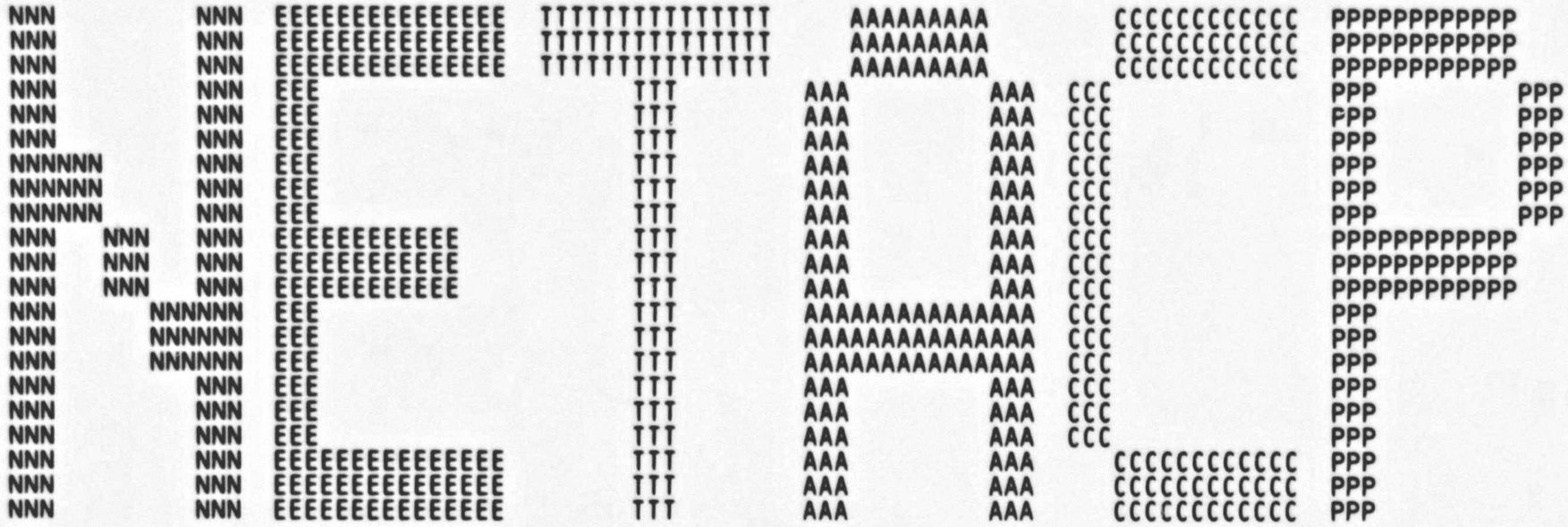
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN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

```
NN      NN  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    LL          LL          TTTTTTTTTT  RRRRRRRR    NN      NN
NN      NN  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    LL          LL          TTTTTTTTTT  RRRRRRRR    NN      NN
NN      NN  EE              TT      DD      DD  LL          LL              TT      RR      RR  NN      NN
NN      NN  EE              TT      DD      DD  LL          LL              TT      RR      RR  NN      NN
NNNN    NN  EE              TT      DD      DD  LL          LL              TT      RR      RR  NNNN    NN
NNNN    NN  EE              TT      DD      DD  LL          LL              TT      RR      RR  NNNN    NN
NN  NN  NN  EEEEEEEE        TT      DD      DD  LL          LL              TT      RRRRRRRR    NN  NN  NN
NN  NN  NN  EEEEEEEE        TT      DD      DD  LL          LL              TT      RRRRRRRR    NN  NN  NN
NN    NNNN  EE              TT      DD      DD  LL          LL              TT      RR  RR      NN    NNNN
NN    NNNN  EE              TT      DD      DD  LL          LL              TT      RR  RR      NN    NNNN
NN      NN  EE              TT      DD      DD  LL          LL              TT      RR    RR    NN      NN      ....
NN      NN  EE              TT      DD      DD  LL          LL              TT      RR    RR    NN      NN      ....
NN      NN  EEEEEEEEEE      TT      DDDDDDDD    LLLLLLLLLL  LLLLLLLLLL      TT      RR      RR  NN      NN      ....
NN      NN  EEEEEEEEEE      TT      DDDDDDDD    LLLLLLLLLL  LLLLLLLLLL      TT      RR      RR  NN      NN      ....

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

Table of contents

(2) 423 Declarations
(3) 570 Macros
(4) 654 Define circuit states
(5) 694 Define circuit transition action routines
(6) 755 Define circuit state table
(7) 833 Define message mapping table
(8) 899 Storage definitions
(9) 1019 NET$INIT_ROUTING - Initialize routing database
(10) 1046 NET$DLLUPDLNI - Process modified LNI parameters
(11) 1156 FORCE_FULL_DECISION - Force full decision algorithm
(12) 1180 NET$DLL_ALL_OFF - Turn off all circuits
(13) 1208 NET$DLL_OPR_SET - Process operator generated event
(14) 1333 ALLOC_LPD - Allocate LPD
(15) 1569 ALLOC_COSTHOPS - Allocate a cost/hops buffer
(16) 1618 DEAL_LPD - Deallocate LPD
(17) 1692 CHECK_REQ_PARAMS - Check that required parameters are set
(18) 1756 NET$DLL_X25_CALL - Process incoming X.25 call
(19) 1836 NET$DLL_X25_RESET - X.25 reset detected
(20) 1886 NET$DLL_RCV - Process message received from driver
(21) 2018 Received message pre-processing routines
(22) 2256 RCV_STR2 - Received Phase II start message
(23) 2341 RCV_STR3 - Received Phase III start message
(24) 2406 RCV_STR4 - Received Phase IV start message
(25) 2461 RCV_VRF - Received routing verification message
(26) 2518 RCV_RHEL - Received Phase IV Router Hello message
(27) 2611 RCV_EHEL - Received Phase IV Endnode Hello message
(28) 2680 RCV_RT3 - Received Phase III routing message
(29) 2755 RCV_RT4 - Received Phase IV routing message
(30) 2821 RCV_ART - Area Routing message received
(31) 2884 Check for routing update loss
(32) 2953 Parse phase II/III/IV address
(33) 3030 PARSE_VERSION - Parse version number field
(34) 3100 SET_DLL_EVT - Schedule event transition
(35) 3126 NET$DLL_PRC_WQE - Process work queue element
(36) 3154 PROC_EVT - Process an event
(37) 3252 FIND_WQE_CTX - Find context for a new WQE
(38) 3319 Simple transition routines
(39) 3377 ACT_RCV_STR - Received start message
(40) 3503 ADAPT_TO_PARTNER - Adapt to partner's node type
(41) 3566 ACT_RCV_VRF - Received verification message
(42) 3675 ACT_RCV_RHEL - Received Router Hello message
(43) 3822 ACT_ELECT - Resolve election after waiting for ballots
(44) 3865 ACT_RCV_EHEL - Received Endnode Hello message
(45) 3915 ACT_RCV_RT - Receive routing message
(46) 4031 UPDATE_MATRIX - Update the routing matrix
(47) 4093 ACT_RCV_ART - Receive area routing message
(48) 4193 REQUEST_UPDATE - Request update of routing database
(49) 4251 UPDATE - Update database and neighbors
(50) 4327 DECISION - Update forwarding database
(51) 4531 FIND_PATH_TO_NODE - Find least cost path to node
(52) 4656 AREA_DECISION - Update area forwarding database
(53) 4761 FIND_PATH_TO_AREA - Find least cost path to area
(54) 4846 UPD_NEIGHBORS - Schedule routing messages
(55) 4937 TIMER_XRT - Automatic routing update timer
(56) 4974 Start automatic routing update timer
(57) 5007 ENDNODE_DECISION - Endnode decision algorithm
(58) 5059 ACT_ENT_MOP - Enter MOP state
(59) 5092 ACT_DLL_UP - Datalink has initialized
(60) 5231 DLE-related state changes
(61) 5319 ACT_RUN_DOWN, ACT_SET_OPER

```
(62)     5393          ACT_TST_DL - Circuit acceptance algorithm
(63)     5452          ACT_ENT_RUN - Enter RUN state
(64)     5529          ACT_BC_UP - Broadcast datalink has initialized
(65)     5609          BRA_UP - Setup new adjacency for BRA
(66)     5726          LOWEST_PRIO_BRA - Find lowest priority BRA
(67)     5766          BEA_UP - Setup new adjacency for BEA
(68)     5854          Error action routines for "RUN" state
(69)     5921          EXIT_RUN_STATE - Exit the RUN state
(70)     6057          ADJ_DOWN - Mark adjacency as shutdown
(71)     6122          BRA_DOWN - Mark BRA down
(72)     6195          BUILD_RTR_LIST - Re-build NI router/state list
(73)     6265          ELECT_ROUTER - Elect designated router
(74)     6318          ACT_QIO_SHUT - Shutdown the datalink
(75)     6406          ACT_QIO_STRT - Start the datalink
(78)     6664          ACT_PVC_START - Start an X.25 PVC in multiple steps
(79)     6754          ACT_X25_CALL - Accept incoming X.25 call
(80)     6799          CHK_CIRC_START - Check if circuit can be started
(81)     6849          TOGGLE_LINE - Shutdown and startup line
(82)     6889          ACT_XMT - Transmit pending messages
(83)     6930          XMT_DALLY - Dally before sending start message
(84)     6967          XMT_STR - Transmit start message
(85)     7086          XMT_VRF - Transmit verification message
(86)     7175          XMT_RT - Transmit a routing message
(87)     7246          XMT_RT4 - Transmit a Phase IV routing message
(88)     7349          XMT_ART - Transmit a Phase IV area routing message
(89)     7455          CHK_IO - Check for multiple transmits
(90)     7488          NET$DLL_QIO_CO - Common QIO routine
(91)     7656          SET_IOTIM - Set I/O timer
(92)     7675          RESET_CHAN - Cancel all device I/O
(93)     7699          NET$GET_LPD_CRI - Locate CNF given LPD index
(94)     7730          NET$ADJ_LPD_CRI - Locate CNF given ADJ index
(95)     7762          NET$LOCATE_LPD - Locate LPD given CNF
(96)     7781          NET$FIND_LPD - Find LPD given LPD index
(97)     7808          NET$FIND_ADJ - Find LPD & ADJ given ADJ index
(98)     7838          NET$GET_PLVECLPD - Find next active LPD
(99)     7870          TELL_NETDRIVER - Inform NETDRIVER of an event
```

```
0000     1          .TITLE  NETDLLTRN  - Routing & Datalink control layer
0000     2          .IDENT  'V04-000'
0000     3          .DEFAULT DISPLACEMENT,LONG
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
0000    29
0000    30  ;++
0000    31  ; FACILITY:     NETWORK ACP
0000    32  ;
0000    33  ; ABSTRACT:
0000    34  ;
0000    35  ;
0000    36  ; ENVIRONMENT:
0000    37  ;
0000    38  ;       Kernel mode
0000    39  ;
0000    40  ; AUTHOR:       A.Eldridge, CREATION DATE: 11-APR-80
0000    41  ;
0000    42  ; MODIFIED BY:
0000    43  ;
0000    44  ;       V040    TMH0040         Tim Halvorsen   20-Jul-1984
0000    45  ;               Fix code which accidentally drops an area routing message
0000    46  ;               on a point-to-point circuit when it is received before the
0000    47  ;               circuit can be changed into the "run" state (this is a
0000    48  ;               race condition, and will occur whenever the remote node
0000    49  ;               is faster, and can send routing messages quickly after
0000    50  ;               node initialization; e.g. RSX).  The symptoms are that
0000    51  ;               the remote area is "unreachable" for up to 3 minutes,
0000    52  ;               even though the circuit has initialized.
0000    53  ;
0000    54  ;       V039    TMH0039         Tim Halvorsen   24-May-1984
0000    55  ;               Remove CANCEL on DLM circuits during shutdown (TMH0036)
0000    56  ;               which turned out to act like a DEACCESS, and removed all
0000    57  ;               X.25 level 3 knowledge of the outstanding reset status,
```

```
0000     58 ;                  putting the circuit into a stuck state.
0000     59 ;                  Only issue an PVC ACCESS once in the life of the circuit,
0000     60 ;                  meaning that circuit re-initialization due to a remote RESET
0000     61 ;                  will just issue a RESET-CONFIRM and transmit a "Start" msg.
0000     62 ;                  Just issue a "reset confirm" when we get notification of an
0000     63 ;                  incoming reset message - the failed receive IRP will be
0000     64 ;                  sufficient mechanism for recycling the circuit.  (This
0000     65 ;                  obsoletes ACT_RUN_RESET).
0000     66 ;                  When exiting the run state for any reason (like either
0000     67 ;                  software error or remote reset), issue a reset message to
0000     68 ;                  the remote side to ensure that it recycles thru node init.
0000     69 ;                  Minimize the computation of the Square Root Limit result with
0000     70 ;                  the maximum allowed value of 127.
0000     71 ;
0000     72 ;          V038    TMH0038         Tim Halvorsen   22-Apr-1984
0000     73 ;                  Wait a few seconds after circuit initialization before
0000     74 ;                  declaring ourself the designated router.  This is so that
0000     75 ;                  we give some time to receive Router Hello messages from
0000     76 ;                  remote nodes which might have a higher priority than us.
0000     77 ;                  When the decision algorithm can't decide between two paths
0000     78 ;                  of equal cost, use the highest node address of the adjacent
0000     79 ;                  nodes as the tiebreaker.
0000     80 ;
0000     81 ;          V037    RNG0037         Rod Gamache     7-Feb-1984
0000     82 ;                  Fix problem where if 2 Level II routers where in different areas
0000     83 ;                  and there are no other nodes in their respective areas, then
0000     84 ;                  the remained "unreachable"! This has to do with the fact that
0000     85 ;                  the RTG_CHG flag never got set, so the decision algorithm was
0000     86 ;                  never run.
0000     87 ;
0000     88 ;          V036    TMH0036         Tim Halvorsen   15-Sep-1983
0000     89 ;                  When a X.25 reset mailbox message is received while doing
0000     90 ;                  PVC_START initialization, ignore it, since PVC_START always
0000     91 ;                  does a "reset confirm" as the last thing it does.  This
0000     92 ;                  prevents a duplicate Reset from being sent, and prevents
0000     93 ;                  aborting the remote side's node initialization if it gets
0000     94 ;                  it after starting node init.
0000     95 ;
0000     96 ;                  When a X.25 reset mailbox message is received during node
0000     97 ;                  init, restart the entire node init process, rather than
0000     98 ;                  ignoring it, since it is possible that a node init message
0000     99 ;                  got lost in the PVC reset.
0000    100 ;
0000    101 ;                  Issue CANCEL on DLM circuits during shutdown to clean up
0000    102 ;                  outstanding PSI requests, such as outstanding resets.
0000    103 ;
0000    104 ;                  Fix code which accidentally drops a routing message on a
0000    105 ;                  point-to-point circuit when it is received before the
0000    106 ;                  circuit can be changed into the "run" state (this is a
0000    107 ;                  race condition, and will occur whenever the remote node
0000    108 ;                  is faster, and can send routing messages quickly after
0000    109 ;                  node initialization; e.g. RSX).. The symptoms are that
0000    110 ;                  the remote node is "unreachable" for up to 3 minutes,
0000    111 ;                  even though the circuit has initialized.
0000    112 ;
0000    113 ;          V035    TMH0035         Tim Halvorsen   11-Jul-1983
0000    114 ;                  Support alias local addresses (cluster addresses) by
```

```
0000   115 ;                  zeroing the cost/hops entry for that address in routing
0000   116 ;                  messages.
0000   117 ;
0000   118 ;          V034    TMH0034         Tim Halvorsen   06-Jun-1983
0000   119 ;                  Detect null passwords from RSX Phase III nodes, which
0000   120 ;                  are sent as 8 bytes of 0, rather than a 0 byte string.
0000   121 ;
0000   122 ;          V033    TMH0033         Tim Halvorsen   25-May-1983
0000   123 ;                  Fix BC circuit rundown so that it calls DLE$BC_DOWN
0000   124 ;                  even though we are an endnode.  Previously, if we were
0000   125 ;                  an endnode, it was skipping the call and leaving the
0000   126 ;                  channels assigned, preventing further service functions.
0000   127 ;                  Use our own node type (in the LPD) to determine what kind
0000   128 ;                  of message to send, rather than using the remote node's type
0000   129 ;                  and jamming his type to support TRANSPORT TYPE.  This is
0000   130 ;                  basically a cleanup of the logic, and should not change
0000   131 ;                  the effective algorithm.
0000   132 ;                  Fix bug in forced phase resynchronization which caused our
0000   133 ;                  side to "act" as the same type as the remote node if we receive
0000   134 ;                  his start msg before successfully initiating the transmission
0000   135 ;                  of our start msg.
0000   136 ;
0000   137 ;          V032    TMH0032         Tim Halvorsen   17-May-1983
0000   138 ;                  Fix bug in version checking in the error path which
0000   139 ;                  crashes the system.
0000   140 ;
0000   141 ;          V031    TMH0031         Tim Halvorsen   06-May-1983
0000   142 ;                  Fix bug in endnode decision algorithm which prevented
0000   143 ;                  endnodes from talking to other endnodes over point-to-point
0000   144 ;                  circuits.
0000   145 ;
0000   146 ;          V030    TMH0030         Tim Halvorsen   26-Apr-1983
0000   147 ;                  Log "dropped by adjacent node" when we receive an
0000   148 ;                  "I'm going away" RHEL from a remote node.
0000   149 ;                  Require verification passwords from Phase III nodes
0000   150 ;                  during point-to-point node init. to prevent accidental
0000   151 ;                  "merging" of area address spaces via an intermediate
0000   152 ;                  Phase III node between them.
0000   153 ;
0000   154 ;          V029    RNG0029         Rod Gamache     20-Apr-1983
0000   155 ;                  Fix branch destinations out of range.
0000   156 ;
0000   157 ;          V028    TMH0028         Tim Halvorsen   06-Apr-1983
0000   158 ;                  Fix code which attempts to ignore Transport data packets
0000   159 ;                  when parsing messages from new adjacencies.
0000   160 ;                  Allow no more than 20 outstanding transmits (of routing msgs)
0000   161 ;                  on an NI circuit, to prevent queue pileup of a stuck datalink,
0000   162 ;                  and to prevent too much storage from being tied up in such a
0000   163 ;                  condition.
0000   164 ;                  Remove some obsolete symbols.
0000   165 ;                  If NBRA is exceeded when trying to add a new BRA, then eject
0000   166 ;                  the lowest priority BRA (rather than simply ignoring the new
0000   167 ;                  BRA).  This way, all BRAs eject the same node from the
0000   168 ;                  "cluster" of BRAs.  Log "adjacency rejected" when a BRA is
0000   169 ;                  thrown out due to the database being full.
0000   170 ;                  Log "address change" reason when a remote adjacency is detected
0000   171 ;                  to have recycled, rather than "listener timeout".
```

```
0000   172 ;
0000   173 ;       V027    TMH0027         Tim Halvorsen   02-Mar-1983
0000   174 ;               Completely rewrite DLE handling.
0000   175 ;               Fix bug which caused BUFFAIL conditions to leave an LPD
0000   176 ;               in the off-synchronizing state if the last message sent
0000   177 ;               to the ACP for the LPD was ignored.
0000   178 ;               Force the cost/hops to infinity when the area becomes
0000   179 ;               unreachable, to speed up process of an area going away.
0000   180 ;               Set flag in RCB if we are allowed to use level 2 routing
0000   181 ;               or not - set false if we detect ourselves to be an isolated
0000   182 ;               area router.
0000   183 ;               Do not allow transport type other than Phase IV on broadcast
0000   184 ;               circuits.
0000   185 ;               Adapt to Phase III endnodes properly by acting as a Phase
0000   186 ;               III router (rather than a Phase III endnode).
0000   187 ;               Change size of hello timer in Start, RHEL and EHEL messages
0000   188 ;               to be a word rather than a byte, and add temporary code to
0000   189 ;               continue to receive messages in the old format (1 byte hello).
0000   190 ;               Remove code to parse/save seed value from messages.  It is
0000   191 ;               never looked at.
0000   192 ;               Send "I'm going away" message (empty RHEL) when a BC circuit
0000   193 ;               is manually turned off.
0000   194 ;               Fix code which attempts to allow more than one area to coexist
0000   195 ;               on the NI by disallowing level2-level1 connections, by dropping
0000   196 ;               level 1 routing messages from other areas even if we're an area
0000   197 ;               router, and by discounting nodes in other areas while electing
0000   198 ;               a designated router.
0000   199 ;               Rather than sending out RHELs every second if there is at least
0000   200 ;               one 1-way connection, send out RHELs only if it has changed
0000   201 ;               since the last time we sent one.  This prevents RHELs every
0000   202 ;               second in the event that a connection is stuck in 1-way mode.
0000   203 ;               Remove incorrect check which required a remote Phase IV router
0000   204 ;               to have it's block size large enough to hold an entire routing
0000   205 ;               message - this check should only be done for Routing III nodes.
0000   206 ;               Remove obsolete XMT_RESTR mechanism - it assumed that incorrect
0000   207 ;               implementations of Phase III nodes could accept a Phase IV
0000   208 ;               start because it looked like a Phase III start.  This is not
0000   209 ;               the case - Phase IV node numbers make them unacceptable to any
0000   210 ;               Phase III node.
0000   211 ;               Dally for a few seconds before sending the initial start msg.
0000   212 ;               This gives us a chance to hear the remote node's start message,
0000   213 ;               and figure out what he is, before sending him a Phase IV msg
0000   214 ;               he may not like (some implementations can't handle Phase IV
0000   215 ;               messages, even to ignore them).
0000   216 ;               Remove code which "remembers" the remote node's type over a
0000   217 ;               circuit recycle.  This was the old method of phase resynch,
0000   218 ;               but was error prone when patching different systems of
0000   219 ;               different types into the same line.
0000   220 ;
0000   221 ;       V026    TMH0026         Tim Halvorsen   24-Jan-1983
0000   222 ;               Fix bug which prevented forced Phase III circuits to
0000   223 ;               endnodes from correctly detecting the remote node as
0000   224 ;               a Phase III endnode.
0000   225 ;               Allow LPD to be set as a Phase III endnode, so that
0000   226 ;               we can adapt to remote Phase III routers as an endnode.
0000   227 ;               This means changing all endnode checks to check for both
0000   228 ;               Phase III and Phase IV endnode node types.
```

```
0000   229 ;               Fix bug in code which tries to prevent sending of routing
0000   230 ;               messages for forced-endnode circuits - wasn't clearing
0000   231 ;               the work flag, and we went into an infinite loop.
0000   232 ;               Init cell in LPD which gives the datalink buffer size
0000   233 ;               and use it throughout, rather than the RCB value.  This
0000   234 ;               is so that datalink buffer sizes can be variable depending
0000   235 ;               on the datalink's route header.
0000   236 ;               Remove any maximum on a circuit's "input packet limiter".
0000   237 ;               Fix bug in computation of "nearest level 2 router" that
0000   238 ;               was causing it to be wiped out on partial decision
0000   239 ;               algorithms.
0000   240 ;               If the MAXIMUM WINDOW parameter is specified, then use it
0000   241 ;               as the input packet limiter for non-X.25 circuits.  This
0000   242 ;               essentially expands the usage of MAXIMUM WINDOW to both
0000   243 ;               X.25 and non-X.25 circuits.
0000   244 ;               Add endnode key support.
0000   245 ;               Prevent transport-type from being set to a router if
0000   246 ;               the executor is set to an endnode.
0000   247 ;               Add code to toggle a DMC line on listener timeouts.
0000   248 ;               Fix forced-phase III circuits to correctly ignore messages
0000   249 ;               with versions higher than phase III, so that node init
0000   250 ;               with higher version nodes works correctly.
0000   251 ;
0000   252 ;       V025    TMH0025         Tim Halvorsen   08-Jan-1983
0000   253 ;               Add Decision Update Vector which gets updated any time
0000   254 ;               a routing message is received which is different than
0000   255 ;               the last one we heard from the same place.  It is used
0000   256 ;               to limit the number of nodes which need to be looked
0000   257 ;               at in the Decision algorithm.
0000   258 ;               Fix loopback check for forced Phase II links to work
0000   259 ;               (it was broken by areas in the local address).
0000   260 ;               Journal new records at the start and finish of the
0000   261 ;               decision algorithm, so routing analysis can be done.
0000   262 ;               Journal all messages written directly to the datalink
0000   263 ;               via QIO from this routine.
0000   264 ;               Remove restriction that prevents a Phase II circuit
0000   265 ;               from initializing if the partner node is already
0000   266 ;               reachable in another part of the network.  This was
0000   267 ;               a restriction needed for the Phase II routing architecture,
0000   268 ;               and is no longer applicable for the current needs of
0000   269 ;               Phase II circuits.
0000   270 ;
0000   271 ;       V024    TMH0024         Tim Halvorsen   17-Dec-1982
0000   272 ;               Re-arrange received message dispatching to locate the
0000   273 ;               CNF, LPD and ADJ blocks before parsing the message.
0000   274 ;               Ignore all messages on NI from another area if we
0000   275 ;               are a level 2 router.
0000   276 ;               Fix circuit re-cycling, so that startup attempts wait
0000   277 ;               for NETDRIVER's IRPCNT in the LPD to go to zero before
0000   278 ;               recycling.  Prevents late breaking CRDs from interrupting
0000   279 ;               the next circuit startup attempt, resulting in TWO line
0000   280 ;               synchronization lost events.
0000   281 ;               Fix calculation of routing update loss for Phase IV
0000   282 ;               route messages, so that it correctly uses the highest
0000   283 ;               reachable node, rather than the highest unreachable
0000   284 ;               node.
0000   285 ;
```

```
0000   286 ;       V023    TMH0023         Tim Halvorsen   01-Dec-1982
0000   287 ;               Disable listen timer for Phase II links, since Phase
0000   288 ;               II didn't have any mandatory hello timer.
0000   289 ;               Make RECALL TIMER parameter work for all types of
0000   290 ;               circuits, to give control over Initialization retry
0000   291 ;               attempts.  Increase size of default interval from 3
0000   292 ;               seconds to 10 seconds to cut down on overhead when
0000   293 ;               a datalink goes down, especially if the datalink is
0000   294 ;               going up and down continuously.
0000   295 ;
0000   296 ;       V022    TMH0022         Tim Halvorsen   13-Oct-1982
0000   297 ;               Select incoming DLM circuits by DTE address,
0000   298 ;               if specified on the incoming circuits.
0000   299 ;               Fix so that a full routing message is sent
0000   300 ;               when a new BRA enters the run state (propagating
0000   301 ;               the database very quickly for the new node).  This
0000   302 ;               is done by the existing BRA, because the new BRA's
0000   303 ;               "request for routing info" (a routing msg) may have
0000   304 ;               been dropped by all routers while waiting for 2-way
0000   305 ;               communication to be established.
0000   306 ;               Add area routing support.
0000   307 ;               Fix bug which prevents values of MAX RECALLS greater
0000   308 ;               than 127 from working (sign bit was being tested).
0000   309 ;               Fix bug in outgoing calls which constructed an illegal
0000   310 ;               PSI NCB if either the WINDOW SIZE or MAXIMUM DATA were
0000   311 ;               specified.
0000   312 ;               Change order of "MOP detected" events, so that line synch.
0000   313 ;               lost comes out before remotely initiated state change.
0000   314 ;               Set the local cost/hops for endnodes to 0, since the
0000   315 ;               decision algorithm is never run (which does it normally).
0000   316 ;               Only reset partner node type every other startup attempt,
0000   317 ;               so that if a startup attempt fails due to wrong version,
0000   318 ;               then the next one will start with the right version.
0000   319 ;               Force the cost/hops to infinity when the node becomes
0000   320 ;               unreachable, to speed up process of node going away.
0000   321 ;
0000   322 ;       V021    TMH0021         Tim Halvorsen   26-Sep-1982
0000   323 ;               Do not allow BLOCKING parameter to be specified,
0000   324 ;               since we don't currently support X.25 blocking.
0000   325 ;               Add endnode support.
0000   326 ;               Fix bug in point-to-point initialization with
0000   327 ;               endnodes.
0000   328 ;
0000   329 ;       V020    TMH0020         Tim Halvorsen   20-Sep-1982
0000   330 ;               Fix "phase resynchronization", so that if we are the
0000   331 ;               higher phase, then process the start message received
0000   332 ;               from the other side (since the lower phase won't ever
0000   333 ;               retransmit it).
0000   334 ;               Allow "late arrival" of start and verification messages
0000   335 ;               to take care of phase resynchronization problems.
0000   336 ;               Fix bug in endnode handling which caused crash when
0000   337 ;               the endnode came up.
0000   338 ;
0000   339 ;       V019    TMH0019         Tim Halvorsen   30-Aug-1982
0000   340 ;               Fix DLE cancel, so that it correctly causes the XWB
0000   341 ;               to be aborted (prevents consistency bugcheck) by using
0000   342 ;               the full path ID, rather than the LPD index in all cases.
```

```
0000   343 ;               Fix transport resynchronization, so that the start message
0000   344 ;               is not reset if we receive a higher phase start, but only
0000   345 ;               if we "lower" our start phase (essentially entering
0000   346 ;               compatibility mode).
0000   347 ;               Fix so that Phase II nodes are not included in Phase IV
0000   348 ;               routing messages (and clean up the code a little by removing
0000   349 ;               Phase II "OL vector" and using the adjacency block instead).
0000   350 ;               Allow Phase II node init from a node address greater
0000   351 ;               than 241 (up to the phase III limit of 255) to allow
0000   352 ;               "forced phase II" gateways using "hidden nodes".
0000   353 ;
0000   354 ;       V018    TMH0018         Tim Halvorsen   02-Jul-1982
0000   355 ;               Don't ever modify the LPD$V_ACTIVE flag - that flag
0000   356 ;               is only to be modified by NETDRIVER, since it is used
0000   357 ;               as a flag to decide whether the IRP_DOWN signal needs
0000   358 ;               to be sent to the ACP or not.
0000   359 ;               Fix bug in code which toggles the line off and on when
0000   360 ;               a fatal controller error is detected by the circuit.
0000   361 ;               It was causing circuits to stay in the synchronizing
0000   362 ;               substate, rather than restarting themselves.
0000   363 ;               Change psect name on DLLTRN state table, so that it is
0000   364 ;               mapped after the main body of the ACP code and data.
0000   365 ;               Add support for broadcast circuits (UNA).
0000   366 ;               Change routines which reference local LPD and which scan
0000   367 ;               LPD vector to use new LPD vector, which is a vector of
0000   368 ;               longword pointers to the actual LPD blocks.  Change the
0000   369 ;               code which allocates LPD slots, to actually allocate an
0000   370 ;               LPD structure from non-paged pool, and insert it's address
0000   371 ;               into the LPD pointer vector.
0000   372 ;               Remove the cost/hops matrix, and instead, allocate a cost/
0000   373 ;               hops buffer for each LPD as it gets initialized, and store
0000   374 ;               the address of each cost/hops buffer into a new vector,
0000   375 ;               based on LPD index.
0000   376 ;               Remove DLL_COST vector, and instead, store and retreive
0000   377 ;               the circuit cost from the LPD block.
0000   378 ;               Add code to support adjacencies in conjunction with NETDRIVER.
0000   379 ;               Add check to ensure that NUMBER is specified with OUTGOING DLM
0000   380 ;               X.25 circuits.
0000   381 ;               Add missing code to pass the maximum window and maximum packet
0000   382 ;               size to PSI when making an outgoing DLM call.
0000   383 ;               Change calling interface to PROC_EVT and DLL_PRC_WQE so that
0000   384 ;               R1 doesn't have to be set to the event longword.
0000   385 ;
0000   386 ;       V017    TMH0017         Tim Halvorsen   28-Jun-1982
0000   387 ;               Enable use of X.25 datagrams by NETDRIVER.
0000   388 ;               Store PSI UCB address in LPD for DLM circuits.
0000   389 ;               Do not touch IOST2 field in X.25 IRP, but assume
0000   390 ;               that the datalink has gone down.
0000   391 ;
0000   392 ;       V016    TMH0016         Tim Halvorsen   25-Mar-1982
0000   393 ;               Fix bug in parsing of node address field for Phase II and
0000   394 ;               Phase III messages.
0000   395 ;               Heavily comment this module and add subtitles.
0000   396 ;               Fix psect naming conventions.
0000   397 ;               Remove all explicit displacement specifiers from operands
0000   398 ;               and make default displacement = word for the entire module.
0000   399 ;               Remove X state, which used to wait for a SHUTDOWN to complete,
```

```
0000   400 ;               but the W state already does this.
0000   401 ;               Remove transition which causes SHUTDOWN to be re-issued if
0000   402 ;               a new access comes in.
0000   403 ;               Get rid of CND_SHUT (action routine 7), change all references
0000   404 ;               of CND_SHUT to a new redefined action routine 2 (CND_STRT),
0000   405 ;               which issues a startup QIO if ASTCNT is zero.
0000   406 ;               Remove IRP_EVT event, which was used to dispatch to IRP_DOWN
0000   407 ;               or IRP_MM, based on the contents of the IRP.  Now, this
0000   408 ;               dispatching is done upon immediately receiving the IRP.
0000   409 ;               Remove obsolete CNF_CRI event - no longer referenced.
0000   410 ;               Get rid of ACT_RCV_STRTIM (action routine 3), and cleanup
0000   411 ;               the timer events to eliminate needless chaining.
0000   412 ;               Cleanup I/O timer code.
0000   413 ;               Add X.25 datalink support.
0000   414 ;               Remove RCV_UNK event, since it didn't do anything.
0000   415 ;               Log "packet format error" if we get an unrecognized message.
0000   416 ;               Change ACT_ENT_MOP to log the event 5.0 or 5.1 (datalink
0000   417 ;               state change) when we go into MOP mode.
0000   418 ;               Log "aborted service request, line open error" if we are
0000   419 ;               unable to create the detached NML process to handle remotely
0000   420 ;               initiated service functions.
0000   421 ;-
```

```
                        .SBTTL  Declarations
                ;
                ; INCLUDE FILES:
                ;
                        $CCBDEF
                        $CNFDEF
                        $CXBDEF
                        $NFBDEF
                        $NMADEF
                        $DDTDEF
                        $DEVTRNDEF
                        $DLLQIODEF
                        $EVCDEF
                        $IRPDEF
                        $LPDDEF
                        $ADJDEF
                        $MSGDEF
                        $NETMSGDEF
                        $NETSYMDEF
                        $NETUPDDEF
                        $NSPMSGDEF              ; DNA architecture definitions
                        $RCBDEF
                        $UCBDEF
                        $WQEDEF
                        $XMDEF
                        $PSIDEF                 ; PSI user definitions (for PSI NCB structure)

                ;
                ;  EQUATED SYMBOLS
                ;
00000000        FDT_LEGAL  = 0                  ; FDT offset to legal functions
00000008        FDT_IOTYPE = 8                  ; FDT offset to function type (buffer/direct)

000000B4        TR$C_TIM_DLLIO  = 3*60          ; ACP datalink I/O timeout period (sec)
0000000A        TR$C_TIM_RESTRT = 10            ; Transport Init retry interval (sec)
00000002        TR$C_TIM_DALLY  = 2             ; Dally for 2 seconds before sending start
                                                ; msg for resynch with dumb Phase III nodes
00000005        TR$C_TIM_DRDELAY = 5            ; Wait at least 5 seconds before declaring
                                                ; ourself "designated router" to give us time
                                                ; to hear from other routers on the NI.

                ;
                ;  Define Phase IV Transport message symbols
                ;
0000000B        TR4C_MSG_RHEL   =  ^B00001011   ; Phase IV Router Hello message
0000001B        TR4C_RHEL_LNG   =          27   ; Length of fixed portion of message
000000EC        TR4C_MAX_RSLIST =         236   ; Maximum size of R/S list
00000000        TR4V_RS_PRIO    =           0   ; Start of router priority field in R/S LIST
00000006        TR4S_RS_PRIO    =           6   ; Length of field
00000007        TR4V_RS_TWOWAY  =           7   ; flag set if 2-way communication with router

0000000D        TR4C_MSG_EHEL   =  ^B00001101   ; Phase IV Endnode Hello message
00000020        TR4C_EHEL_LNG   =          32   ; Length of fixed portion of message

00000001        TR4C_MSG_STR    =  ^B00000001   ; Start message type code
0000000C        TR4C_STR_LNG    =          12   ; Fixed start message length
00000000        TR4V_REQ_NTY    =           0   ; Start of TLINFO field specifying node type
```

```
00000002  0000   480 TR4S_REQ_NTY    =          2   ; Length of the field
00000001  0000   481 TR4C_NTY_ARO    =          1   ; Field value for area routing nodes
00000002  0000   482 TR4C_NTY_ROU    =          2   ; Field value for routing nodes
00000003  0000   483 TR4C_NTY_NROU   =          3   ; Field value for non-routing nodes
00000002  0000   484 TR4V_REQ_VRF    =          2   ; TIINFO bit - set if verification is requested
          0000   485
00000003  0000   486 TR4C_MSG_VRF    =  ^B00000011  ; Verification message type code
00000004  0000   487 TR4C_VRF_LNG    =          4   ; Length of fixed portion of verfication msg
00000044  0000   488 TR4C_VRF_MXL    =         68   ; Verification message max length
00000040  0000   489 TR4C_MAX_PSW    =         64   ; Maximum password text length
          0000   490
00000009  0000   491 TR4C_MSG_ART    =  ^B00001001  ; Area routing message type code
00000006  0000   492 TR4C_ART_LNG    =          6   ; Length of fixed portion of routing message
          0000   493
00000007  0000   494 TR4C_MSG_RT     =  ^B00000111  ; Routing message type code
00000006  0000   495 TR4C_RT_LNG     =          6   ; Length of fixed portion of routing message
00000000  0000   496 TR4V_RT_COST    =          0   ; Begining of COST field
0000000A  0000   497 TR4S_RT_COST    =         10   ; Size of COST field
0000000A  0000   498 TR4V_RT_HOPS    =         10   ; Begining of HOPS field
00000005  0000   499 TR4S_RT_HOPS    =          5   ; Size of HOPS field
          0000   500
00000006  0000   501 TR4C_MSG_ENH    =  ^B00000110  ; Phase IV endnode data packet - always ignored here
          0000   502
00000002  0000   503 TR4C_TIVER      =    ^X000002  ; Phase IV version = 2.0.0
00000002  0000   504 TR4C_T3MULT     =          2   ; Hello/listen factor for non-broadcast circuits
00000003  0000   505 TR4C_BCT3MULT   =          3   ; Hello/listen factor for broadcast circuits
          0000   506
          0000   507 ;
          0000   508 ;  Define Phase III Transport message symbols
          0000   509 ;
00000001  0000   510 TR3C_MSG_STR    =  ^B00000001  ; Start message type code
0000000A  0000   511 TR3C_STR_LNG    =         10   ; Fixed start message length
00000009  0000   512 TR3C_STR_RSXL   =          9   ;!RSX work around
00000000  0000   513 TR3V_REQ_NTY    =          0   ; Start of TLINFO field specifying node type
00000002  0000   514 TR3S_REQ_NTY    =          2   ; Length of the field
00000002  0000   515 TR3C_NTY_PH3    =          2   ; Field value for routing nodes
00000003  0000   516 TR3C_NTY_PH3N   =          3   ; Field value for non-routing nodes
00000002  0000   517 TR3V_REQ_VRF    =          2   ; TLINFO bit - set if verification is requested
          0000   518
00000003  0000   519 TR3C_MSG_VRF    =  ^B00000011  ; Verification message type code
00000004  0000   520 TR3C_VRF_LNG    =          4   ; Length of fixed portion of verfication msg
00000044  0000   521 TR3C_VRF_MXL    =         68   ; Verification message max length
00000040  0000   522 TR3C_MAX_PSW    =         64   ; Maximum password text length
          0000   523
00000007  0000   524 TR3C_MSG_RT     =  ^B00000111  ; Routing message type code
00000005  0000   525 TR3C_RT_LNG     =          5   ; Length of fixed portion of routing message
00000000  0000   526 TR3V_RT_COST    =          0   ; Begining of COST field
0000000A  0000   527 TR3S_RT_COST    =         10   ; Size of COST field
0000000A  0000   528 TR3V_RT_HOPS    =         10   ; Begining of HOPS field
00000005  0000   529 TR3S_RT_HOPS    =          5   ; Size of HOPS field
          0000   530
00000002  0000   531 TR3C_MSG_RTH    =  ^B00000010  ; Phase III/IV data packet - always ignored here
          0000   532
00000301  0000   533 TR3C_TIVER      =    ^X000301  ; Phase III version = 1.3.0
00000005  0000   534 TR3C_MSG_TST    =  ^B00000101  ; Test (hello) message type code
          0000   535 ;!TR3C_TST_MAX  =         128   ; Maximum size of test data field
0000007F  0000   536 TR3C_TST_MAX    =         127   ;!RSX work-around
```

```
00000003  0000   537 TR3C_NUM_TST    =          3   ; Number of test messages to send during
          0000   538                                ; acceptance testing.
          0000   539 ;
          0000   540 ;  Define Phase II Transport message symbols
          0000   541 ;
000000FF  0000   542 TR2C_MAX_PNA    =        255   ; Maximum Phase II partner node address
          0000   543
00000008  0000   544 TR2C_MSG_NOP    =  ^B00001000   ; NOP message type code
00000001  0000   545 TR2C_NOP_LNG    =          1   ; Mininum NOP message length
00000000  0000   546 TR2C_NUM_NOP    =          0   ; Number of NOP message to send to test the
          0000   547                                ; circuit during initialization
          0000   548
00000058  0000   549 TR2C_MSG_INI    =  ^B01011000   ; Initialization message type code
          0000   550
00000001  0000   551 TR2C_INI_STR    =  ^B00000001   ; Initialization start sub-type code
0000000A  0000   552 TR2C_STR_LNG    =         10   ; Length of fixed portion of start message  ;!
00000050  0000   553 TR2C_STR_MXL    =         80   ; Max length of start message               ;!
00000000  0000   554 TR2C_STR_FCT    =  ^B00000000   ; Expected start message "function" field value
00000006  0000   555 TR2C_STR_REQ    =  ^B00000110   ; Expected start message "request" field value
00000000  0000   556 TR2V_REQ_VRF    =          0   ; "request" field modifier to request a
00000001  0000   557 TR2M_REQ_VRF    =      ^X<01>  ;   verification message
00000002  0000   558 TR2M_FCT_INT    =      ^X<02>  ; "function" field modifier to show that the
          0000   559                                ; node does intercept functions
          0000   560
00000002  0000   561 TR2C_INI_VRF    =  ^B00000010   ; Initialization verification sub-type code
00000002  0000   562 TR2C_VRF_LNG    =          2   ; Length of verf msg minus password length
00000008  0000   563 TR2C_PSW_LNG    =          8   ; Length of verf msg password
          0000   564 ;
          0000   565 ;  Define common Routing constants
          0000   566 ;
00000044  0000   567 TR_C_VRF_LNG    = TR3C_VRF_MXL  ; Maximum verification msg size
00000040  0000   568 TR_C_MAX_PSW    = TR3C_MAX_PSW  ; Maximum size of verification password
```

```
                                                H 3
0000    570          .SBTTL  Macros
0000    571
0000    572 ;
0000    573 ;  MACROS
0000    574 ;
0000    575 .MACRO  $LOG  code,qual1,qual2,reg                  ; Setup logging info
0000    576
0000    577                         _$log = evc$c_'code'
0000    578          .IIF NB,qual1,  _$log = _$log + <<evc$c_'qual1'>@16>
0000    579          .IIF NB,qual2,  _$log = _$log + <<evc$c_'qual2'>@24>
0000    580
0000    581          MOVL    #_$log,WQE$W_EVL_CODE(reg)
0000    582 .ENDM   $LOG
0000    583
0000    584
0000    585 .MACRO  $DSP_TABLE  list                            ; Setup dispatch table
0000    586
0000    587          .MACRO  $dspent _$dspinx,_$dspact
0000    588
0000    589                  .IIF GT,  <_$dspinx-_$maxinx>,  _$maxinx = _$dspinx
0000    590                  .         = _$tmp + <4 * _$dspinx>
0000    591                  .address   _$dspact
0000    592          .ENDM   $dspent
0000    593
0000    594          _$tmp    = .
0000    595          _$maxinx = 0
0000    596          .IRP    a,<LIST>
0000    597          $dspent a
0000    598          .ENDR
0000    599
0000    600 . = _$tmp + <4 * _$maxinx> + 4
0000    601
0000    602 .ENDM   $DSP_TABLE
0000    603
0000    604
00000010 0000   605 LEV$C_STATES   = 16                          ; Number of columns in the table
FFFFFFFF 0000   606 LEV$C_MAX_EVT  = -1                         ; Init the number of rows
0000    607
0000    608 .MACRO  $LEV  event, s, w,y, m, a,b,c,d,j, r        ; Create state table entries
0000    609                                                     ; for the specified circuit event
0000    610
0000    611          LEV$C_MAX_EVT = LEV$C_MAX_EVT + 1          ; Bump max event value
0000    612          LEV$C_'event' == LEV$C_MAX_EVT             ; Define circuit event symbol
0000    613
0000    614                  $ENT    s,_s                       ; Create table entry
0000    615
0000    616                  $ENT    w,_w
0000    617                  $ENT    y,_y
0000    618
0000    619                  $ENT    m,_m
0000    620
0000    621                  $ENT    a,_a
0000    622                  $ENT    b,_b
0000    623                  $ENT    c,_c
0000    624                  $ENT    d,_d
0000    625                  $ENT    j,_j
0000    626
```

```
0000   627                  $ENT    r,_r
0000   628
0000   629                  $ENT    ?,_s                    ; Pad so that each row in the
0000   630                  $ENT    ?,_s                    ; table is a multiple of 16
0000   631                  $ENT    ?,_s                    ; so that the state table
0000   632                  $ENT    ?,_s                    ; journal file is easy to read
0000   633                  $ENT    ?,_s
0000   634                  $ENT    ?,_s
0000   635 .ENDM   $LEV
0000   636
0000   637
0000   638
0000   639 .MACRO  $ENT    entry,def_sta                   ; Create state table entry
0000   640
0000   641                   $ent = %LENGTH(entry)-1
0000   642                  lev$c_sta_. = lev$c_sta'def_sta'; Define default next state
0000   643
0000   644          .IF IDN,entry,?                         ; ? => bug
0000   645              .BYTE  lev$c_sta_.                  ; Use current state
0000   646              .BYTE  4                            ; Action is bug-check
0000   647          .IFF
0000   648              .BYTE  lev$c_sta_%EXTRACT(0,1,entry); Setup next state
0000   649              .BYTE  %EXTRACT(T,_$ent,entry)      ; Setup action routine index
0000   650
0000   651          .ENDC
0000   652 .ENDM   $ENT
```

```
                                                J 3
0000    654          .SBTTL  Define circuit states
0000    655 ;
0000    656 ;  Circuit LPD States (LPD$B_STI values)
0000    657 ;
0000    658 $EQULST LEV$C_STA_,,0,1,<-;
0000    659
0000    660          <S>     -; Stopping: There is an active channel to the device but has
0000    661                  -;           either been stopped or has been given a command to
0000    662                  -;           stop. There may be timer or I/O ast's pending.
0000    663                  -;
0000    664                  -;
0000    665                  -;   DATA LINK LAYER INITIALIZATION OR RESTART
0000    666                  -;
0000    667          <W>     -; Shutting: A shutdown QIO was issued, and completion pending.
0000    668                  -;           The device has been given a command to shutdown so
0000    669                  -;           that it is in a known state prior to being started.
0000    670                  -;
0000    671          <Y>     -; Starting: A startup QIO was issued, and completion pending.
0000    672                  -;
0000    673                  -;
0000    674                  -;   NORMAL MAINTAINANCE MODE STATE (MOP MODE)
0000    675                  -;
0000    676          <M>     -; Maintenance: In use by another process for service functions
0000    677                  -;
0000    678                  -;
0000    679                  -;   TRANSPORT LAYER INITIALIZATION
0000    680                  -;
0000    681          <A>     -; Waiting for:  xmt idle, rcv verf, rcv init
0000    682          <B>     -; Waiting for:  xmt idle, rcv verf
0000    683          <C>     -; Waiting for:  xmt idle
0000    684          <D>     -; Waiting for:            rcv verf
0000    685                  -;
0000    686          <J>     -; Undergoing circuit acceptance testing
0000    687                  -;
0000    688                  -;
0000    689                  -;   NORMAL RUNNING STATES
0000    690                  -;
0000    691          <R>     -; Running:  Available for normal traffic.
0000    692 >
```

```
                        .SBTTL  Define circuit transition action routines

                        .PSECT  NET_PURE,NOWRT,NOEXE,LONG

LEV_AL_ACTTAB:

                    $DSP_TABLE  -
                    <-
                        < 0, ACT_NOP>           -; Nop action routine
                        <32, ACT_EXIT>          -; Exit state table processing
                        < 4, ACT_BUG>           -; Bugcheck
                        <11, ACT_NYI>           -; Not yet implemented
                        < 2, ACT_QIO_STRT>      -; If LPD$B_ASTCNT=0,
                                                -; Issue startup QIO, reset i/o timer
                        <34, ACT_X25_CALL>      -; Accept incoming X.25 call on circuit
                        <42, ACT_X25_RESET>     -; Respond to X.25 reset during initialization
                        <17, ACT_PVC_START>     -; Startup PVC in multiple steps
                        < 1, ACT_QIO_SHUT>      -; Issue shutdown QIO, reset i/o timer
                        < 6, ACT_RUN_SYNC>      -; Synchronization lost in run state
                        < 8, ACT_RUN_UXPK>      -; Unexpected packet rcv'd in run state
                        <16, ACT_RUN_SHUT>      -; Shut down from RUN state
                        <39, ACT_ADJ_DOWN>      -; Mark adjacency down
                                                -;
                        <18, ACT_DLL_UP>        -; The datalink has initialized, begin next
                                                -; phase (Transport or DLE) of activity
                        <10, ACT_ENT_RUN>       -; Enter RUN state
                        <22, ACT_ENT_MPR>       -; Circuit entered MOP mode while in RUN state
                        < 9, ACT_ENT_MOP>       -; Circuit entered MOP mode
                        <26, ACT_ENT_DLE>       -; The circuit has become available for use by
                                                -; a server process for direct-line access
                        <37, ACT_BC_UP>         -; A broadcast circuit has initialized
                                                -;
                        <19, ACT_XMT>           -; Send a message if possible
                        <30, ACT_RCV_2STR>      -; Respond to second rcvd "start" msg
                        <12, ACT_RCV_STR>       -; Respond to rcvd "start" msg
                        <13, ACT_RCV_VRF>       -; Respond to rcvd "verification" msg
                        <20, ACT_RCV_RT>        -; Respond to rcvd Routing msg
                        <29, ACT_RCV_RTA>       -; Receive Routing msg while acceptance testing
                        <35, ACT_RCV_ART>       -; Respond to rcvd area routing msg
                        <36, ACT_RCV_ARTA>      -; Receive area routing msg while testing
                        <40, ACT_RCV_RHEL>      -; Respond to rcvd "Router Hello" msg
                        <41, ACT_RCV_EHEL>      -; Respond to rcvd "Endnode Hello" msg
                                                -;
                        <44, ACT_ELECT>         -; Elect 1st "designated router"
                                                -;
                        <43, ACT_FAILED>        -; Mark a circuit "failed"
                        < 5, ACT_RUN_DOWN>      -; Cancel all timers, etc.
                        <31, ACT_SET_OPER>      -; Simulate a "set operators state" event
                        <15, ACT_EXI_SERV>      -; Exit service state if needed
                                                -;
                        <21, ACT_TST_DL>        -; Run acceptance algorithm
                        <23, ACT_REQ_UPDATE>    -; Request routing database update
                                                -;
                        <25, ACT_LOG_NFE>       -; Log event
                        <24, ACT_LOG_CDE>       -; Log event & shutdown circuit
                        <38, ACT_LOG_ADE>       -; Log event & shutdown adjacency
                                                -;
```

```
0000   751          <27, ACT_SYN_FAIL>      -; The circuit failed to synchronize
0000   752          <28, ACT_INI_FAIL>      -; I/O failure during transport initialization
0000   753     >
```

```
                        .SBTTL  Define circuit state table

                        .SAVE_PSECT
00000000                .PSECT  TABLES_PURE,NOEXE,NOWRT,GBL     ; Separate psect for ease
                                                                ; of journalling display

                LEV$AW_STA_TAB:
                ;
                ;                       S   W   Y     M     A   B   C   D   J     R
                ;               --------+---+---+-----+-----+---+---+---+---+-----+--
0000    765 $LEV    NO_EVT      .5  .32 .32   .19   .19 .19 .19 .32 .21   .19
0020    766 $LEV    EXIT        :   :   :     :     :   :   :   :   :     :
0040    767 $LEV    BUG         ?   ?   ?     .     ?   ?   ?   ?   ?     ?
0060    768
0060    769 $LEV    UNJAM       .   W1  W1    W1    W1  W1  W1  W1  W1    W6
0080    770 $LEV    REQ_SHUT    .1  W1  W1    W1    W1  W1  W1  W1  W1    W1
00A0    771
00A0    772 $LEV    OPR_OFF     .   S27 S27   S27   S1  S1  S1  S1  S1    W6
00C0    773 $LEV    OPR_ON      W1  .15 .15   .15   .   .   .   .   .     .23
00E0    774 $LEV    OPR_SRV     W1  .   .     .     W1  W1  W1  W1  W1    W6
0100    775
0100    776 $LEV    RCV_STR     .   .   .     .     B12 B30 B30 B30 B30   .8
0120    777 $LEV    RCV_VRF     .   .   .     .     .   .13 W1  .13 W1    .8
0140    778 $LEV    RCV_VVF     .   .   .     .     ?   C19 ?   J21 W1    ?
0160    779 $LEV    RCV_RT      .   .   .     .     .   .   .   .   .29   .20
0180    780 $LEV    RCV_ART     .   .   .     .     .   .   .   .   .36   .35
01A0    781 $LEV    RCV_RHEL    .   .   .     .     .   .   .   .   .     .40
01C0    782 $LEV    RCV_EHEL    .   .   .     .     .   .   .   .   .     .41
01E0    783
01E0    784 $LEV    XMT_IDLE    .   .   .     .     ?   D   J21 .32 R10   .32
0200    785
0200    786 $LEV    LIN_UP      .   .   A18   .     ?   ?   ?   ?   R10   .
0220    787 $LEV    LIN_DOWN    .   .   W27   W27   W1  W1  W1  W1  W1    W16
0240    788 $LEV    ADJ_DOWN    .   .   W27   W27   W1  W1  W1  W1  W1    .39
0260    789
0260    790 $LEV    BC_UP       ?   ?   R37   ?     ?   ?   ?   ?   ?     ?
0280    791
0280    792 $LEV    IO_TIMOUT   .31 .27 W27   .     W27 W28 W28 W28 W28   W6
02A0    793 $LEV    IO_FAIL     .   Y2  W27   W27   W27 W28 W28 W28 W28   W6
02C0    794 $LEV    IO_SUCC     .   Y2  A18   .9    .   .   .   ?   .     .
02E0    795
02E0    796 $LEV    X25_CALL    .   Y34 ?     ?     ?   ?   ?   ?   ?     ?
0300    797 $LEV    PVC_START   .   .   .17   ?     ?   ?   ?   ?   ?     ?
0320    798 $LEV    X25_RESET   .   Y42 .     .     .42 .42 .42 .42 .42   .42
0340    799
0340    800 $LEV    STRT_TIM    .   Y2  .     .     .   .   .   .   .     .
0360    801 $LEV    ELECT_TIM   .   .   .     .     .   .   .   .   .     .44
0380    802
0380    803 $LEV    FAILED      ?   S43 S43   ?     ?   ?   ?   ?   ?     ?
03A0    804
03A0    805 $LEV    ENT_DLE     ?   ?   M26   ?     ?   ?   ?   ?   ?     ?
03C0    806 $LEV    DLE_ACC     W1  W1  W1    .26   W1  W1  W1  W1  W1    W16
03E0    807
03E0    808 $LEV    IRP_RESET   .   Y2  W27   W27   W27 W28 W28 W28 W28   W6
0400    809 $LEV    IRP_DOWN    .   Y2  W27   W27   W27 W28 W28 W28 W28   W6
0420    810 $LEV    IRP_MM      S9  S9  S9    W27   S9  S9  S9  S9  S9    W22
0440    811
```

```
                     0440   812 $LEV  LOG_NFE     .25 .25 .25   .25   .25 .25 .25 .25 .25   .25
                     0460   813 $LEV  LOG_CDE     .24 .24 .24   .24   .24 .24 .24 .24 .24   .24
                     0480   814 $LEV  LOG_ADE     .38 .38 .38   .38   .38 .38 .38 .38 .38   .38
                     04A0   815
          00000025   04A0   816 LEV$C_EVENTS  = LEV$C_MAX_EVT+1
                     04A0   817
                 000000B4   818         .RESTORE_PSECT
                     00B4   819
                     00B4   820 ;
                     00B4   821 ; NOTE: Action routines which are dispatched to upon the LEV$C_NO_EVT event
                     00B4   822 ;       must not exit with the LEV$C_NO_EVT; if no events are to be chained
                     00B4   823 ;       to then these routines must exit with LEV$C_EXIT.  Failure to adhere
                     00B4   824 ;       to this rule will result in an infinite loop in the state table.
                     00B4   825 ;
                     00B4   826 ;       If an action routine is never dispatched to upon the LEV$C_NO_EVT
                     00B4   827 ;       event then it must never exit with LEV$C_EXIT; if no events are to
                     00B4   828 ;       be chained to then these routines must exit with LEV$C_NO_EVT.
                     00B4   829 ;       Failure to adhere to this rule could result in failure to deallocate
                     00B4   830 ;       an LPD which is no longer needed.
                     00B4   831 ;
```

```
                                                B 4
00B4   833          .SBTTL  Define message mapping table
00B4   834 ;
00B4   835 ;  Define message mapping table
00B4   836 ;
00B4   837 .MACRO  MSGTAB    parser,min_siz,msg_typ
00B4   838
00B4   839          .ADDRESS  parser
00B4   840          .WORD     min_siz
00B4   841          .WORD     msg_typ
00B4   842
00B4   843 .ENDM   MSGTAB
00B4   844 ;
00B4   845 MSG_MAP_TABLE:
00B4   846
00B4   847      MSGTAB      RCV_STR2,  TR2C_STR_LNG,  <<TR2C_INI_STR@8>!TR2C_MSG_INI>
00BC   848      MSGTAB      RCV_VRF2,  TR2C_VRF_LNG,  <<TR2C_INI_VRF@8>!TR2C_MSG_INI>
00C4   849      MSGTAB      RCV_STR3,  TR3C_STR_RSXL,                  TR3C_MSG_STR
00CC   850      MSGTAB      RCV_VRF3,  TR3C_VRF_LNG,                   TR3C_MSG_VRF
00D4   851      MSGTAB      RCV_RT,    TR3C_RT_LNG,                    TR3C_MSG_RT
00DC   852      MSGTAB      RCV_ART,   TR4C_ART_LNG,                   TR4C_MSG_ART
00E4   853      MSGTAB      RCV_RHEL,  TR4C_RHEL_LNG,                  TR4C_MSG_RHEL
00EC   854      MSGTAB      RCV_EHEL,  TR4C_EHEL_LNG,                  TR4C_MSG_EHEL
00F4   855      MSGTAB      0,         0,                  0
00FC   856
00FC   857
00FC   858 ;
00FC   859 ;  Setup mapping from CRI states to operator events
00FC   860 ;
00FC   861          ASSUME  NMA$C_STATE_ON  EQ 0
00FC   862          ASSUME  NMA$C_STATE_OFF EQ 1
00FC   863          ASSUME  NMA$C_STATE_SER EQ 2
00FC   864
06     00FC   865 OPR_EVT_MAP:     .BYTE   LEV$C_OPR_ON
05     00FD   866                  .BYTE   LEV$C_OPR_OFF
07     00FE   867                  .BYTE   LEV$C_OPR_SRV
00     00FF   868                  .BYTE   0
       0100   869
       0100   870 ;
       0100   871 ; Define "destination NI addresses" for broadcast QIOs issued here
       0100   872 ;
       0100   873
       0100   874 NET$G_ALL_ROU:
030000AB 0100 875          .LONG   TR$C_NI_ALLROU1          ; Multicast = "all routers"
    0000 0104 876          .WORD   TR$C_NI_ALLROU2
       0106   877
       0106   878 ;
       0106   879 ; Define a CRC polyonomial table to compute CRC-16 checksums
       0106   880 ;
       0106   881
00000000 0106 882 CRC16:   .LONG   ^X00000000
0000CC01 010A 883          .LONG   ^X0000CC01
0000D801 010E 884          .LONG   ^X0000D801
00001400 0112 885          .LONG   ^X00001400
0000F001 0116 886          .LONG   ^X0000F001
00003C00 011A 887          .LONG   ^X00003C00
00002800 011E 888          .LONG   ^X00002800
0000E401 0122 889          .LONG   ^X0000E401
```

```
0000A001  0126   890          .LONG   ^X0000A001
00006C00  012A   891          .LONG   ^X00006C00
00007800  012E   892          .LONG   ^X00007800
0000B401  0132   893          .LONG   ^X0000B401
00005000  0136   894          .LONG   ^X00005000
00009C01  013A   895          .LONG   ^X00009C01
00008801  013E   896          .LONG   ^X00008801
00004400  0142   897          .LONG   ^X00004400
```

```
                     0146   899          .SBTTL  Storage definitions
                     0146   900
                 00000000   901          .PSECT  NET_IMPURE,WRT,NOEXE,LONG
                     0000   902 ;
                     0000   903 ;  Define miscellaneous storage
                     0000   904 ;
          00000000   0000   905 NET$GL_INITVER::.LONG   0        ; For saving received Init Message version
00000000  00000000   0004   906 LEV_Q_CRI:      .QUAD   0        ; For saving CRI CNF and CNR
          00000000   000C   907 LEV_L_LPD:      .LONG   0        ; For saving LPD address
          00000000   0010   908 LEV_L_ADJ:      .LONG   0        ; For saving ADJ address
          00000000   0014   909 LEV_W_PNA:      .LONG   0        ; For saving partner's address
          00000000   0018   910 LEV_W_BLKSIZE:  .LONG   0        ; Partner's receive block size
          00000000   001C   911 LEV_B_PRIORITY: .LONG   0        ; Partner's router priority
          00000000   0020   912 LEV_W_HELLO:    .LONG   0        ; Partner's hello timer
00000000  00000000   0024   913 LEV_Q_PSWDESC:  .QUAD   0        ; For saving descriptor of rcvd password
          00000000   002C   914 MAX_HOPS:       .LONG   0        ; Max total hops allowed
          00000000   0030   915 MAX_COST:       .LONG   0        ; Max total path cost allowed
          00000000   0034   916 XMTFLG:         .LONG   0        ; For LPD$B_XMTFLG image
          00000000   0038   917 PTYPE:          .LONG   0        ; Type of partner node (routing, endnode, etc.)
          00000000   003C   918 NULL:           .LONG   0        ; For dummy node name
              0000   0040   919 RTGFLG:         .WORD   0        ; Routing flags
          00000000   0042   920 RTG_V_RUS = 0                    ; Update supression timer is ticking
          00000001   0042   921 RTG_V_UPD = 1                    ; Request was made to run "update"
                     0042   922
                 00000146   923          .PSECT  NET_PURE,NOWRT,NOEXE,LONG
                     0146   924
                     0146   925 ;
                     0146   926 ; Maximum value allowed for computed Square Root Limit (SRL).
                     0146   927 ;
          0000007F   0146   928 MAX_SRL:        .LONG   127      ; Maximum signed byte value
                     014A   929
                     014A   930 ;
                     014A   931 ; Table to convert partner type codes into a "phase" designation
                     014A   932 ; (i.e. Phase II, Phase III, etc.) to be used in transport re-synchronization.
                     014A   933 ;
                     014A   934
                     014A   935          .MACRO  PHDEF   PTY,PHASE
                     014A   936          .SAVE_PSECT
                     014A   937          . = PTY_TO_PHASE + PTY
                     014A   938          .BYTE   PHASE
                     014A   939          .RESTORE_PSECT
                     014A   940          .ENDM
                     014A   941
                     014A   942 PTY_TO_PHASE:
          00000154   014A   943          .BLKB   10                      ; Allocate table of 10 cells
                     0154   944                                          ; and fill them in with:
                     0154   945          PHDEF   ADJ$C_PTY_PH2,2
                     0154   946          PHDEF   ADJ$C_PTY_PH3,3
                     0154   947          PHDEF   ADJ$C_PTY_PH3N,3
                     0154   948          PHDEF   ADJ$C_PTY_PH4,4
                     0154   949          PHDEF   ADJ$C_PTY_PH4N,4
                     0154   950          PHDEF   ADJ$C_PTY_AREA,4
                     0154   951
                     0154   952
                     0154   953 ;
                     0154   954 ; Table to convert partner type codes into a version number to be used in
                     0154   955 ; message version checking.
```

```
                     0154   956 ;
                     0154   957
                     0154   958          .MACRO  VERDEF  PTY,VERS
                     0154   959          .SAVE_PSECT
                     0154   960          . = PTY_TO_VERSION + <2*PTY>
                     0154   961          .WORD   VERS
                     0154   962          .RESTORE_PSECT
                     0154   963          .ENDM
                     0154   964
                     0154   965 PTY_TO_VERSION:
00000168             0154   966          .BLKW   10                      ; Allocate table of 10 cells
                     0168   967                                          ; and fill them in with:
                     0168   968          VERDEF  ADJ$C_PTY_PH2,0
                     0168   969          VERDEF  ADJ$C_PTY_PH3,TR3C_TIVER
                     0168   970          VERDEF  ADJ$C_PTY_PH3N,TR3C_TIVER
                     0168   971          VERDEF  ADJ$C_PTY_PH4,TR4C_TIVER
                     0168   972          VERDEF  ADJ$C_PTY_PH4N,TR4C_TIVER
                     0168   973          VERDEF  ADJ$C_PTY_AREA,TR4C_TIVER
                     0168   974
                     0168   975
00000000             0168   976          .PSECT  TABLES_IMPURE,NOEXE,WRT,GBL,LONG
                     0000   977
00000400             0000   978 NUM_NODES = NET$C_MAX_NODES + 1          ; Use zero indexed structures
00000041             0000   979 NUM_CIRCS = NET$C_MAX_LINES + 1
00000040             0000   980 NUM_AREAS = NET$C_MAX_AREAS + 1
                     0000   981
                     0000   982
                     0000   983 REACH_EVT:
00000080             0000   984          .BLKB   <NUM_NODES+7>/8          ; Bit vector used to monitor
                     0080   985                                          ; node reachability changes
                     0080   986
                     0080   987 RTG_CHG:
00000100             0080   988          .BLKB   <NUM_NODES+7>/8          ; Bit vector used to monitor
00000080             0100   989 RTG_CHG_LEN = .-RTG_CHG                   ; node routing info changes
                     0100   990
                     0100   991          .ALIGN WORD
                     0100   992
                     0100   993 NET$AW_MIN_C_H::
00000900             0100   994          .BLKW   NUM_NODES                ; Minimum Cost/Hops vector
                     0900   995
                     0900   996 NET$AW_AREA_C_H::
00000980             0900   997          .BLKW   NUM_AREAS                ; Area Minimum Cost/Hops vector
                     0980   998
                     0980   999          .ALIGN LONG
                     0980  1000
                     0980  1001 NET$AL_CH_VEC::
00001A88             0980  1002          .BLKL   1+NUM_CIRCS+NUM_NODES
                     1A88  1003                                          ; Vector of addresses of buffers
                     1A88  1004                                          ; which hold the last levl 1 routing message
                     1A88  1005                                          ; received from the circuit or BRA.
                     1A88  1006                                          ; NUM_NODES should be enough space for
                     1A88  1007                                          ; the maximum broadcast routers.
                     1A88  1008
                     1A88  1009 NET$AL_AREA_CH::
00002B90             1A88  1010          .BLKL   1+NUM_CIRCS+NUM_NODES
                     2B90  1011                                          ; Vector of addresses of buffers
                     2B90  1012                                          ; which hold the last area routing message
```

```
                2B90  1013                                    ; received from the circuit or BRA.
                2B90  1014                                    ; NUM_NODES should be enough space for
                2B90  1015                                    ; the maximum broadcast routers.
                2B90  1016
            00000000  1017          .PSECT  NET_CODE,NOWRT,EXE
```

```
                                  0000          .SBTTL  NET$INIT_ROUTING - Initialize routing database
                                  0000  ;+
                                  0000  ; NET$INIT_ROUTING - Initialize routing database
                                  0000  ;
                                  0000  ; This routine is called when the ACP is starting up, to initialize
                                  0000  ; any routing database that needs it.
                                  0000  ;
                                  0000  ; Inputs:
                                  0000  ;
                                  0000  ;       None
                                  0000  ;
                                  0000  ; Outputs:
                                  0000  ;
                                  0000  ;       R0 = Status code
                                  0000  ;
                                  0000  ;       All other registers are destroyed.
                                  0000  ;-
                                  0000  NET$INIT_ROUTING::
                                  0000          ;
                                  0000          ;   Initialize the minimum cost/hops vector
                                  0000          ;
0080 8F   FF 8F   6E   00    2C   0000          MOVC5   #0,(SP),#-1,#2*NUM_AREAS,NET$AW_AREA_C_H ; Min. area cost/hops
             00000900'EF          0008
0800 8F   FF 8F   6E   00    2C   000D          MOVC5   #0,(SP),#-1,#2*NUM_NODES,NET$AW_MIN_C_H  ; Min. cost/hops vector
             00000100'EF          0015
                    000D     30   001A          BSBW    FORCE_FULL_DECISION     ; Force full decision algorithm
                 50   00'    D0   001D          MOVL    S^#SS$_NORMAL,R0        ; Success
                             05   0020          RSB
```

```
                                                H  4
                     0021  1046          .SBTTL  NET$DLLUPDLNI - Process modified LNI parameters
                     0021  1047 ;+
                     0021  1048 ; NET$DLLUPDLNI - Process modified LNI parameters
                     0021  1049 ;
                     0021  1050 ; FUNCTIONAL DESCRIPTION:
                     0021  1051 ;
                     0021  1052 ; This routine is called from module NETACPTRN whenever the LNI data base has
                     0021  1053 ; been updated.
                     0021  1054 ;
                     0021  1055 ; INPUTS:        None
                     0021  1056 ;
                     0021  1057 ; OUTPUTS:       R0      Status code
                     0021  1058 ;
                     0021  1059 ;
                     0021  1060 ;                R1 is destroyed.
                     0021  1061 ;-
                     0021  1062          .SAVE_PSECT
                00000000  1063          .PSECT  NET_LOCK_CODE,NOWRT,EXE,GBL
                     0000  1064
                     0000  1065 NET$DLLUPDLNI::                          ; Update datalink control layer
                     0000  1066 UPDATE_ALL:                              ; Update all routing databases
              0CFC 8F    BB  0000  1067          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R10,R11>; Save regs
    54  00000000'EF    D0  0004  1068          MOVL    NET$GL_PTR_VCB,R4       ; Get RCB pointer
                     000B  1069          ;
                     000B  1070          ;   Calculate the maximum datalink queue length.  The formula
                     000B  1071          ;   dictated by the Transport Archictecture is the number of
                     000B  1072          ;   buffers divided by the square root of the number of circuits.
                     000B  1073          ;
          50  5D A4  9A  000B  1074          MOVZBL  RCB$B_MAX_SNK(R4),R0    ; Setup R0 in case no active circuits
          51  60 A4  9A  000F  1075          MOVZBL  RCB$B_ACT_DLL(R4),R1    ; Get total number of active circuits
                20  13  0013  1076          BEQL    20$                     ; Done if EQL
        52  0082 C4  3C  0015  1077          MOVZWL  RCB$W_MAX_PKT(R4),R2    ; Get the total number of buffers
             52 52  C4  001A  1078          MULL    R2,R2                   ; Square it
             52 51  C6  001D  1079          DIVL    R1,R2                   ; Divide by the number of circuits
             50 01  D0  0020  1080          MOVL    #1,R0                   ; Establish tentative value
             51 01  D0  0023  1081          MOVL    #1,R1                   ; Square of this value
             52 51  D1  0026  1082 10$:     CMPL    R1,R2                   ; Compare square of value to
                     0029  1083                                             ; (buffs**2)/circuits
                0A  1E  0029  1084          BGEQU   20$                     ; If GEQU then we're done
             51 50  C0  002B  1085          ADDL    R0,R1                   ; Begin (n+1)**2 calculation
                50  D6  002E  1086          INCL    R0                      ;  n = n+1
             51 50  C0  0030  1087          ADDL    R0,R1                   ; (n+1)**2 = n**2 + 2*n + 1
                F1  11  0033  1088          BRB     10$
                     0035  1089 20$:    ;
                     0035  1090          ;   Validate maximum allowed value for SRL (Square Root Limit)
                     0035  1091          ;
00000146'EF     50  D1  0035  1092          CMPL    R0,MAX_SRL              ; Is value too large?
                07  1B  003C  1093          BLEQU   25$                     ; Br if no, okay
50  00000146'EF    D0  003E  1094          MOVL    MAX_SRL,R0              ; Else, set to maximum allowed
                     0045  1095 25$:    ;
                     0045  1096          ;   Update the maximum output queue lengths
                     0045  1097          ;
          51  5D A4  9A  0045  1098          MOVZBL  RCB$B_MAX_SNK(R4),R1    ; Get old max queue length
          5D A4  50  90  0049  1099          MOVB    R0,RCB$B_MAX_SNK(R4)    ; Update it
             50 51  C2  004D  1100          SUBL    R1,R0                   ; Get difference (could be negative)
          55  5C A4  9A  0050  1101          MOVZBL  RCB$B_MAX_LPD(R4),R5    ; Get number of cells
                21  13  0054  1102          BEQL    50$                     ; If EQL then none
```

```
                                0056          DSBINT  #NET$C_IPL                  ; Synch with NETDRIVER
        56    28 B445    D0     005C  30$:    MOVL    @RCB$L_PTR_LPD(R4)[R5],R6   ; Get LPD address
                   0E    18     0061          BGEQ    40$                         ; Branch if none in this slot
                10 A6    D5     0063          TSTL    LPD$L_UCB(R6)               ; Any datalink associated with this LPD?
                   09    13     0066          BEQL    40$                         ; Branch if not (local LPD)
                   04    E1     0068          BBC     #LPD$V_RUN,-                ;
             04 22 A6           006A                   LPD$W_STS(R6),40$           ; Adjust only if in run state
          1E A6    50    80     006D          ADDB    R0,LPD$B_XMT_SRL(R6)        ; Adjust square root limiter (could go
                                0071                                              ; negative until some I/O completes!)
                E8 55    F5     0071  40$:    SOBGTR  R5,30$                      ; Loop for each cell
                                0074          ENBINT                              ; Restore IPL
                                0077  50$:    ;
                                0077          ;   On all routing circuits, force a routing message to
                                0077          ;   be sent next time around.
                                0077          ;
          55    5C A4    9A     0077          MOVZBL  RCB$B_MAX_LPD(R4),R5        ; Get number of circuits
        56    28 B445    D0     007B  60$:    MOVL    @RCB$L_PTR_LPD(R4)[R5],R6   ; Get LPD address
                   16    18     0080          BGEQ    65$                         ; Branch if slot not valid
       11 22 A6    04    E1     0082          BBC     #LPD$V_RUN,LPD$W_STS(R6),65$ ; Branch if circuit inactive
        57    2C B445    D0     0087          MOVL    @RCB$L_PTR_ADJ(R4)[R5],R7   ; Get ADJ address
          08 67    02    E1     008C          BBC     #ADJ$V_RTG,ADJ$B_STS(R7),65$ ; Branch if non-routing partner
                                0090          ASSUME  LPD$C_SRM_SIZE EQ 32
          56 A6    01    CE     0090          MNEGL   #1,LPD$G_SRM(R6)            ; Force rtginfo for all nodes to be sent
                                0094          ASSUME  LPD$C_ASRM_SIZE EQ 1        ; && fix this
          5E A6    01    CE     0094          MNEGL   #1,LPD$G_ASRM(R6)           ; Force area rtginfo to all level 2 nodes
                E0 55    F5     0098  65$:    SOBGTR  R5,60$                      ; Loop through all circuits
                                009B          ;
                                009B          ;   Re-run the decision algorithm in 1 second, and send routing
                                009B          ;   messages to our routing neighbors.
                                009B          ;
        51    0202 8F    3C     009B          MOVZWL  #<<WQE$C_QUAL_RTG>@8>!-     ; Set timer ID
                                00A0                  NET$C_TID_XRT,R1
      52    00000021'EF    9E     00A0          MOVAB   UPDATE_TIMER,R2             ; Setup action routine address
53  00000000 00989680 8F    7D     00A7          MOVQ    #10*1000*1000,R3            ; Timer = 1 second
                  FF4B'    30     00B2          BSBW    WQE$RESET_TIM               ; Set the timer ticking
                50    01    D0     00B5  90$:    MOVL    #1,R0                       ; Indicate success
                0CFC 8F    BA     00B8  100$:   POPR    #^M<R2,R3,R4,R5,R6,R7,R10,R11>; Restore regs
                           05     00BC          RSB
                                00BD
                           00000021            .RESTORE_PSECT
                                0021
                                0021  ;
                                0021  ; This timer routine is called to run the decision algorithm.  It
                                0021  ; is done on a timed basis, to avoid sending routing messages in
                                0021  ; the above routine.
                                0021  ;
                                0021
                                0021  UPDATE_TIMER:
                  0D64    30     0021          BSBW    KILL_WQE                    ; Deallocate timer WQE
                    04    10     0024          BSBB    FORCE_FULL_DECISION         ; Force full decision algorithm
                  1404    30     0026          BSBW    REQUEST_UPDATE              ; Request decision
                           05     0029          RSB
```

```
                                  002A          .SBTTL  FORCE_FULL_DECISION - Force full decision algorithm
                                  002A  ;+
                                  002A  ; FORCE_FULL_DECISION - Force decision algorithm to be run on all nodes
                                  002A  ;
                                  002A  ; This routine is called whenever any routing related parameters have
                                  002A  ; changes which might affect cost/hop calculations.
                                  002A  ;
                                  002A  ; Inputs:
                                  002A  ;
                                  002A  ;       None
                                  002A  ;
                                  002A  ; Outputs:
                                  002A  ;
                                  002A  ;       None
                                  002A  ;
                                  002A  ;       No registers are destroyed.
                                  002A  ;-
                                  002A  FORCE_FULL_DECISION:
                        3F    BB  002A          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers
          FF 8F    6E   00    2C  002C          MOVC5   #0,(SP),#-1,-           ; Store 1's in bitvector
  00000080'EF    0080 8F        0031                  #RTG_CHG_LEN,RTG_CHG
                        3F    BA  0039          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers
                              05  003B          RSB
```

```
                                003C  1180          .SBTTL  NET$DLL_ALL_OFF - Turn off all circuits
                                003C  1181 ;+
                                003C  1182 ; NET$DLL_ALL_OFF - Turn off all circuits
                                003C  1183 ;
                                003C  1184 ; FUNCTIONAL DESCRIPTION:
                                003C  1185 ;
                                003C  1186 ; Each CRI entry is forced to the OFF state and an operator event is generated.
                                003C  1187 ;
                                003C  1188 ; INPUTS:        None
                                003C  1189 ;
                                003C  1190 ; OUTPUTS:       All registers are destroyed
                                003C  1191 ;
                                003C  1192 ;-
                                003C  1193 NET$DLL_ALL_OFF::                            ; Turn off all circuits
      54   00000000'EF    D0   003C  1194          MOVL    NET$GL_PTR_VCB,R4           ; Get the RCB address
           55    5C A4    9A   0043  1195          MOVZBL  RCB$B_MAX_LPD(R4),R5        ; Get number of cells
                  24    13   0047  1196          BEQL    50$                         ; If EQL then none
                58   55    D0   0049  1197 30$:     MOVL    R5,R8                       ; Get LPD i.d.
                   2BF9    30   004C  1198          BSBW    NET$GET_LPD_CRI             ; Get LPD and CRI blocks
                  18 50    E9   004F  1199          BLBC    R0,40$                      ; If LBC then not active
                58   01    D0   0052  1200          MOVL    #NMA$C_STATE_OFF,R8         ; Setup new state value
                           0055  1201          $PUTFLD cri,l,sta                   ; Stuff it into the CRI CNF
           50   00FC'C8    9A   0062  1202          MOVZBL  OPR_EVT_MAP(R8),R0          ; Get corresponding event
                   0CF1    30   0067  1203          BSBW    SET_DLL_EVT                 ; Queue the event - always succeeds
                  DC 55    F5   006A  1204 40$:     SOBGTR  R5,30$                      ; Loop for each cell
                50   00'    D0   006D  1205 50$:     MOVL    S^#SS$_NORMAL,R0            ; Indicate success
                           05   0070  1206          RSB
```

```
                                0071          .SBTTL  NET$DLL_OPR_SET - Process operator generated event
                                0071  ;+
                                0071  ; NET$DLL_OPR_SET - Setup operator generated event
                                0071  ;
                                0071  ; FUNCTIONAL DESCRIPTION:
                                0071  ;
                                0071  ; The CRI has been updated and is about to be inserted in the database.  Since
                                0071  ; the circuit "state" may have changed, schedule an event.
                                0071  ;
                                0071  ; INPUTS:        R11     CRI root block pointer
                                0071  ;                R10     CRI block pointer
                                0071  ;                R9      Scratch
                                0071  ;                R8      Value of <cri,l,sta> (operater state)
                                0071  ;                R7-R0   Scratch
                                0071  ;
                                0071  ; OUTPUTS:       R11,R10  are preserved
                                0071  ;                R0      Low bit set if successful
                                0071  ;                        VMS status code otherwise (R9 = Field ID in error)
                                0071  ;
                                0071  ;                All other registers are destroyed.
                                0071  ;
                                0071  ;-
                                0071  NET$DLL_OPR_SET::                          ; Setup operator generated event
                  58    DD      0071          PUSHL   R8                        ; Save state
        0000000C'EF     D4      0073          CLRL    LEV_L_LPD                 ; No LPD allocated yet
        00000000'EF     16      0079          JSB     NET$GET_VEC               ; Prepare the line
              2F 50     E9      007F          BLBC    R0,2$                     ; If LBC then error
  54    00000000'EF     D0      0082          MOVL    NET$GL_PTR_VCB,R4         ; Get RCB address
     05    008A C4      91      0089          CMPB    RCB$B_ETY(R4),#ADJ$C_PTY_PH4N ; If endnode,
                  09    13      008E          BEQL    1$                        ; branch
        00000000'EF     16      0090          JSB     NET$GET_RTG               ; Get routing info
              18 50     E9      0096          BLBC    R0,2$                     ; Branch if error
               2C00     30      0099  1$:     BSBW    NET$LOCATE_LPD            ; Locate associated LPD
                                009C                                            ; R6 = 0 on return if none
        00000000'EF     16      009C          JSB     NET$GET_VEC3              ; Check the line state
              0C 50     E9      00A2          BLBC    R0,2$                     ; If LBC then error
           01    6E     91      00A5          CMPB    (SP),#NMA$C_STATE_OFF     ; Is the STATE OFF ?
                  0D    12      00A8          BNEQ    5$                        ; If NEQ then no
                  56    D5      00AA          TSTL    R6                        ; Is there an LPD
                  06    12      00AC          BNEQ    3$                        ; If NEQ yes, generate state table event
               009E     31      00AE          BRW     90$                       ; Else just return
               009E     31      00B1  2$:     BRW     100$                      ; Exit
               008A     31      00B4  3$:     BRW     80$                       ; Generate state table event
                                00B7  5$:     ;
                                00B7          ;   If STATE is ON, then ensure that all required parameters are set
                                00B7          ;
           00    6E     91      00B7          CMPB    (SP),#NMA$C_STATE_ON      ; Is new STATE ON ?
                  09    12      00BA          BNEQ    10$                       ; If no, then skip checks
               041F     30      00BC          BSBW    CHECK_REQ_PARAMS          ; Ensure required parameters are set
              03 50     E8      00BF          BLBS    R0,10$                    ; Branch if ok
               008D     31      00C2  109$:   BRW     100$                      ; Exit with error
                                00C5          ;
                                00C5          ;   Allocate an LPD, if one does not already exist for this circuit.
                                00C5          ;
                  56    D5      00C5  10$:    TSTL    R6                        ; Is there an LPD ?
                  0D    12      00C7          BNEQ    20$                       ; If NEQ then yes
               009C     30      00C9          BSBW    ALLOC_LPD                 ; Else allocate one
```

```
                F3 50    E9  00CC  1265          BLBC    R0,109$                 ; If LBC then failed
0000000C'EF      56    D0  00CF  1266          MOVL    R6,LEV_L_LPD            ; Save new LPD pointer
                             00D6  1267 20$:     ;
                             00D6  1268          ;   Validate circuit parameters with the datalink driver in
                             00D6  1269          ;   order to return any simple errors immediately to the user.
                             00D6  1270          ;   All errors after this point will simply leave the circuit
                             00D6  1271          ;   in an "on-synchronizing" state.
                             00D6  1272          ;
                             00D6  1273          ;   This is not done for X.25 datalinks, since they don't have
                             00D6  1274          ;   any concept of parameter validation without starting the circuit.
                             00D6  1275          ;
          13 22 A6    07    E0  00D6  1276          BBS     #LPD$V_X25,LPD$W_STS(R6),50$ ; Skip if X.25 datalink
                             00DB  1277          $CNFFLD cri,s,chr,R9            ; Identify characteristics buffer
             52   14 A6    3C  00E2  1278          MOVZWL  LPD$W_CHAN(R6),R2       ; Get I/O channel
                    51    D4  00E6  1279          CLRL    R1                      ; Clear illegal I/O modifier mask
                  FF15'   30  00E8  1280          BSBW    NET$SET_QIOW            ; Get buffer and issue $QIOW
                 64 50    E9  00EB  1281          BLBC    R0,100$                 ; If LBC then failed
                             00EE  1282 50$:     ;
                             00EE  1283          ;   Store cost associated with this circuit
                             00EE  1284          ;
                             00EE  1285          $GETFLD cri,l,cos               ; Get circuit cost
                             00FB  1286          ;                               ; (must be specified at this point)
             29 A6    58    90  00FB  1287          MOVB    R8,LPD$B_COST(R6)       ; Update circuit's cost
                             00FF  1288          ;
                             00FF  1289          ;   Store our NI router priority for this circuit
                             00FF  1290          ;
                             00FF  1291          $GETFLD cri,l,rpr               ; Get NI router priority
                             010C  1292          ;                               ; (if not set, default it to zero)
             2A A6    58    90  010C  1293          MOVB    R8,LPD$B_BCPRI(R6)      ; Store router priority
                             0110  1294          ;
                             0110  1295          ;   Save hello timer in LPD
                             0110  1296          ;
                             0110  1297          $GETFLD cri,l,het               ; Get the hello timer
                   06 50     E8  011D  1298          BLBS    R0,60$                  ; If LBS then it parameter was found
                58    0D     D0  0120  1299          MOVL    #13,R8                  ; Else set the default
                    FEDA'   30  0123  1300          BSBW    CNF$PUT_FIELD           ; Store it in the CRI
          18 A6     58     B0  0126  1301 60$:     MOVW    R8,LPD$W_INT_TLK(R6)    ; Setup the talker interval
                             012A  1302          ;
                             012A  1303          ;   Store X.25 BLOCKING flag into LPD
                             012A  1304          ;
                             012A  1305          $GETFLD cri,l,blk               ; Get BLOCKING parameter
                   07 50     E9  0137  1306          BLBC    R0,80$                  ; If not specified, leave flag=off
                             013A  1307 ;&&      BLBC    R8,80$                  ; Branch if parameter off
                             013A  1308 ;&&      SETBIT  LPD$V_X25BLK,LPD$W_STS(R6) ; Set flag value into LPD
50           0000'8F     3C  013A  1309          MOVZWL  #SS$_BADPARAM,R0        ;&& We don't currently support this
                    11    11  013F  1310          BRB     100$                    ; Exit with error
                             0141  1311          ;
                             0141  1312          ;   Force full decision algorithm to be run on all nodes, in case
                             0141  1313          ;   the cost has changed for this circuit.
                             0141  1314          ;
                    FEE6     30  0141  1315 80$:     BSBW    FORCE_FULL_DECISION     ; Force full decision algorithm
                             0144  1316          ;
                             0144  1317          ;   Generate an event to drive the circuit's state table
                             0144  1318          ;
             58     6E     9A  0144  1319          MOVZBL  (SP),R8                 ; Get new STATE value
50           00FC'C8     9A  0147  1320          MOVZBL  OPR_EVT_MAP(R8),R0      ; Get corresponding event
                  0C0C     30  014C  1321          BSBW    SET_DLL_EVT             ; Queue the event - always succeeds
```

```
                    014F  1322
     50   00'  D0  014F  1323 90$:    MOVL    S^#SS$_NORMAL,R0        ; Indicate success
          8E   D5  0152  1324 100$:   TSTL    (SP)+                   ; Cleanup stack
       10 50   E8  0154  1325         BLBS    R0,110$                 ; Exit if success
0000000C'EF    D5  0157  1326         TSTL    LEV_L_LPD               ; Was LPD just allocated ?
          08   13  015D  1327         BEQL    110$                    ; If EQL then no
          50   DD  015F  1328         PUSHL   R0                      ; Remember status
        02E9   30  0161  1329         BSBW    DEAL_LPD                ; Deallocate the LPD
          50 8ED0  0164  1330         POPL    R0                      ; Restore status
             05  0167  1331 110$:   RSB
```

```
                                                B 5
                     0168  1333          .SBTTL  ALLOC_LPD - Allocate LPD
                     0168  1334 ;+
                     0168  1335 ; ALLOC_LPD     - Allocate and initialize an LPD cell
                     0168  1336 ;
                     0168  1337 ; FUNCTIONAL DESCRIPTION:
                     0168  1338 ;
                     0168  1339 ; A free LPD cell is allocated and initialized.  A channel is assigned to it.
                     0168  1340 ;
                     0168  1341 ;
                     0168  1342 ; INPUTS:        R11     CRI CNR address
                     0168  1343 ;                R10     CRI CNF address
                     0168  1344 ;                R9-R0   Scratch
                     0168  1345 ;
                     0168  1346 ; OUTPUTS:       R11,R10 Unchanged
                     0168  1347 ;                R8      Assigned path i.d.
                     0168  1348 ;                R6      Path's LPD address
                     0168  1349 ;                R0      Low bit set if path was found (or assigned)
                     0168  1350 ;                        Low bit clear otherwise (R9 = field ID in error)
                     0168  1351 ;-
                     0168  1352 ALLOC_LPD:                                  ; Allocate/init an LPD cell
54   00000000'EF  D0 0168  1353          MOVL    NET$GL_PTR_VCB,R4          ; Get RCB address
          56     D4 016F  1354          CLRL    R6                         ; Mark no LPD allocated yet
                     0171  1355          ;
                     0171  1356          ;   Find a free LPD cell
                     0171  1357          ;
  55   5C A4     9A 0171  1358          MOVZBL  RCB$B_MAX_LPD(R4),R5       ; Get max path index
     58   01     D0 0175  1359          MOVL    #1,R8                      ; Start at beginning of vector
53   28 B448     D0 0178  1360 110$:    MOVL    @RCB$L_PTR_LPD(R4)[R8],R3  ; Get LPD address for this index
        0D     18 017D  1361          BGEQ    130$                       ; Branch if index not in use
        58     D6 017F  1362          INCL    R8                         ; Advance to next slot
      F4 55    F5 0181  1363          SOBGTR  R5,110$                    ; Loop
50   0000'8F     3C 0184  1364          MOVZWL  #SS$_INSFMEM,R0            ; Indicate failure
      0222     31 0189  1365 119$:    BRW     300$                       ; Take common exit
                     018C  1366          ;
                     018C  1367          ;   Allocate an LPD block from non-paged pool
                     018C  1368          ;
51   0000006A 8F   D0 018C  1369 130$:   MOVL    #LPD$C_LENGTH,R1           ; Set length of LPD block
           FE6A'   30 0193  1370          BSBW    NET$ALONPGD_Z              ; Allocate LPD block
         F0 50     E9 0196  1371          BLBC    R0,119$                    ; Branch if unable to allocate
      56    52     D0 0199  1372          MOVL    R2,R6                      ; Point to new LPD block
   28 B448    56     D0 019C  1373          MOVL    R6,@RCB$L_PTR_LPD(R4)[R8] ; Mark the slot in use
                     01A1  1374          ;
                     01A1  1375          ;   Allocate a buffer from ACP process space to hold the last
                     01A1  1376          ;   routing message received over this circuit.
                     01A1  1377          ;
                     01A1  1378          $DISPATCH RCB$B_ETY(R4),TYPE=B,<- ; If we are an endnode,
                     01A1  1379                  <ADJ$C_PTY_PH4N,135$>,- ; skip the following
                     01A1  1380                  <ADJ$C_PTY_PH3N,135$>>
         020D     30 01B1  1381          BSBW    ALLOC_COSTHOPS             ; Allocate cost/hops buffer
       D2 50      E9 01B4  1382          BLBC    R0,119$                    ; Branch if error detected
                     01B7  1383 135$:    ;
                     01B7  1384          ;   Initialize the LPD cell
                     01B7  1385          ;
20 A6  53    0100 8F   A1 01B7  1386          ADDW3   #^X<0100>,R3,LPD$W_PTH(R6) ; Set the new path ID
                     01BE  1387                                             ; (increment sequence number)
     1F A6     01   90 01BE  1388          MOVB    #1,LPD$B_XMT_IPL(R6)       ; Setup input packet limiter
          5D A4    90 01C2  1389          MOVB    RCB$B_MAX_SNR(R4),-        ;
```

```
          1E A6        01C5                    LPD$B_XMT_SRL(R6)        ; Setup square root limiter
                       01C7          ASSUME    LPD$Q_REQ_WAIT EQ 0      ;
       66    56   D0   01C7          MOVL      R6,(R6)                  ; Init the queue header
    04 A6    56   D0   01CA          MOVL      R6,4(R6)                 ;
  00000000'GF    D0   01CE          MOVL      G^EXE$GL_ABSTIM,-        ;
          36 A6        01D4                    LPD$L_ABS_TIM(R6)        ; Time counters were zeroed
       004C 8F    A3   01D6          SUBW3     #CXB$C_OVERHEAD,-        ; Default datalink buffer size
          7E A4        01DA                    RCB$W_TOTBUFSIZ(R4),-    ; in case PLVEC doesn't exist
          50 A6        01DC                    LPD$W_BUFSIZ(R6)
                       01DE          ;
                       01DE          ;   Determine if this is an X.25 datalink mapping circuit,
                       01DE          ;   and if so, mark it as such.
                       01DE          ;
                       01DE          $GETFLD   cri,l,typ                ; Get TYPE parameter
          0A 50   E9   01EB          BLBC      R0,140$                  ; Branch if not specified
       03    58   D1   01EE          CMPL      R8,#NMA$C_CIRTY_X25      ; X.25 circuit?
             05   12   01F1          BNEQ      140$                     ; Branch if not
                       01F3          SETBIT    LPD$V_X25,LPD$W_STS(R6)  ; Mark the circuit as X.25 datalink
                       01F8  140$:   ;
                       01F8          ;   Locate the line entry, by searching the line database
                       01F8          ;   looking for a line with the same VMS initial device name.
                       01F8          ;   In order to handle drivers which clone UCBs on each assign
                       01F8          ;   (and so, we can't assign another channel for the circuit
                       01F8          ;   without getting another UCB), we get the actual device name
                       01F8          ;   used by the line (with the cloned unit number filled in)
                       01F8          ;   and assigning a channel to that UCB for the circuit.
                       01F8          ;
                       01F8          ;   In addition, as long as we are looking for the associated
                       01F8          ;   line, copy the line's datalink buffer size to the LPD for
                       01F8          ;   easier access.  If it isn't found for some reason, the LPD
                       01F8          ;   buffer size has been setup earlier with a default value.
                       01F8          ;
    50   0000'8F   3C   01F8          MOVZWL    #SS$_NOSUCHDEV,R0        ; Assume error if bad line name
                       01FD          $GETFLD   cri,s,vmsnam             ; Get device name descriptor
          03 50   E8   020A          BLBS      R0,142$                  ; Br on success
           019E   31   020D          BRW       300$                     ; Else, take common exit
  47 22 A6    07   E0   0210  142$:  BBS       #LPD$V_X25,LPD$W_STS(R6),149$ ; No line for X25 circuits
       0C00 8F    BB   0215          PUSHR     #^M<R10,R11>             ; Save CNR/CNF pointers
5B  00000000'EF   D0   0219          MOVL      NET$GL_CNR_PLI,R11       ; Point to line database
             5A   D4   0220          CLRL      R10                      ; Start at beginning
                       0222          $SEARCH   eql,pli,s,vmsnam         ; Search for line with the same device
          21 50   E9   0231          BLBC      R0,145$                  ; Error if no corresponding line
                       0234          $GETFLD   pli,l,bus                ; Get datalink buffer size
          04 50   E9   0241          BLBC      R0,144$                  ; Skip if not returned
    50 A6    58   B0   0244          MOVW      R8,LPD$W_BUFSIZ(R6)      ; Save datalink buffer size
                       0248  144$:   $GETFLD   pli,s,devnam             ; Get actual device name for line
       0C00 8F    BA   0255  145$:   POPR      #^M<R10,R11>             ; Restore registers
          59 50   E9   0259          BLBC      R0,155$                  ; Branch if error detected
                       025C  149$:   ;
                       025C          ;   Assign a channel to the device.
                       025C          ;
       7E    57   7D   025C          MOVQ      R7,-(SP)                 ; Save name descriptor
       50    5E   D0   025F          MOVL      SP,R0                    ; $ASSIGN_S modifies the SP
                       0262          $ASSIGN_S  -                       ; Get a channel to the device
                       0262                    DEVNAM = (R0),-
                       0262                    CHAN   = LPD$W_CHAN(R6),-
                       0262                    MBXNAM = NET$GQ_MBX_NAME  ; For PSI UCBs, get all mbx msgs
```

```
                                0276                                                    ; (For other UCBs, this does nothing)
                    8E    7C    0276          CLRQ    (SP)+                             ; Cleanup the stack
                 3A 50    E9    0278          BLBC    R0,155$                           ; Br on error
                                027B          ;
                                027B          ;   Find associated LINE (PLVEC) with this device UCB and claim it
                                027B          ;
            50    14 A6    3C    027B          MOVZWL  LPD$W_CHAN(R6),R0                 ; Get channel for call
          00000000'GF      16    027F          JSB     G^IOC$VERIFYCHAN                  ; Get the CCB, ignore errors --
                                0285                                                    ; CCB is returned anyway
            50    61       D0    0285          MOVL    CCB$L_UCB(R1),R0                  ; Get the UCB pointer
          10 A6    50       D0    0288          MOVL    R0,LPD$L_UCB(R6)                  ; Setup the UCB pointer
       78 22 A6    07       E0    028C          BBS     #LPD$V_X25,LPD$W_STS(R6),170$ ; X.25 datalink has no PLVEC
    58  00000000'EF          9A    0291          MOVZBL  PLVEC$GB_MAX,R8                   ; Get max PLVEC index
  50  00000000'EF48          D1    0298  150$:   CMPL    PLVEC$AL_UCB[R8],R0               ; Is this it ?
                    16       13    02A0          BEQL    160$                              ; If EQL then yes
                 F3 58       F5    02A2          SOBGTR  R8,150$                           ; Else loop (index 0 is not used)
                                02A5          $DASSGN_S  CHAN = LPD$W_CHAN(R6); Deassign the channel
          50    0000'8F    3C    02B0          MOVZWL  #SS$_NOSUCHDEV,R0                 ; Indicate error
                  00F6       31    02B5  155$:   BRW     300$                              ; Take common exit
          00000000'EF48      96    02B8  160$:   INCB    PLVEC$AB_REFC[R8]                 ; Another PLVEC cell reference
          28 A6    58       90    02BF          MOVB    R8,LPD$B_PLVEC(R6)                ; Setup PLVEC index
                                02C3          ;
                                02C3          ;   For point-to-point pseudo UNA datalink, always use the same
                                02C3          ;   channel for both line and circuit, so that shared PID/CHAN
                                02C3          ;   matching works in the UNA driver.
                                02C3          ;
  0A  00000000'EF48          91    02C3          CMPB    PLVEC$AB_DEV[R8],-                ; Point-to-point pseudo UNA datalink?
                                02CB                  #DEVTRN$C_DEV_PPUNA
                    19       12    02CB          BNEQ    161$                              ; If so,
                    50       DD    02CD          PUSHL   R0                                ; Save datalink UCB address
                                02CF          $DASSGN_S CHAN = LPD$W_CHAN(R6) ; Deassign the channel done above
                    50 8ED0       02DA          POPL    R0                                ; Restore UCB address for later on
14 A6  00000000'EF48          B0    02DD          MOVW    PLVEC$AW_CHAN[R8],-               ; Use the line's channel
                                02E6                  LPD$W_CHAN(R6)                    ; for the circuit as well
                                02E6  161$:   ;
                                02E6          ;   If the associated line is of PROTOCOL NI, then mark the LPD
                                02E6          ;   as a broadcast circuit and set a flag forcing all I/O to be
                                02E6          ;   word aligned.
                                02E6          ;
  09  00000000'EF48          91    02E6          CMPB    PLVEC$AB_DEV[R8],#DEVTRN$C_DEV_UNA      ; UNA?
                    0A       12    02EE          BNEQ    165$                              ; Branch if not
                                02F0          SETBIT  #LPD$V_BC,LPD$W_STS(R6) ; Mark as broadcast circuit
                                02F5          SETBIT  #LPD$V_ALIGNW,LPD$W_STS(R6) ; Always word-align UNA I/O
                                02FA  165$:   ;
                                02FA          ;   If the associated line is of PROTOCOL CI, then set a flag
                                02FA          ;   forcing all I/O to be quadword aligned.
                                02FA          ;
  04  00000000'EF48          91    02FA          CMPB    PLVEC$AB_DEV[R8],#DEVTRN$C_DEV_CI       ; CI?
                    05       12    0302          BNEQ    166$                              ; Branch if not
                                0304          SETBIT  #LPD$V_ALIGNQ,LPD$W_STS(R6) ; Always quadword-align the CI
                                0309  166$:   ;
                                0309          ;   Determine whether the datalink driver can support buffered
                                0309          ;   or direct I/O based on it's FDT table.
                                0309          ;
          50    0088 C0    D0    0309  170$:   MOVL    UCB$L_DDT(R0),R0                  ; Get DDT address
            50    08 A0    D0    030E          MOVL    DDT$L_FDT(R0),R0                  ; Get FDT
          00000000'8F      E1    0312          BBC     #IO$_READLBLK,-                   ; If BC then direct I/O function
```

```
                05 08 A0        0318  1504                    FDT_IOTYPE(R0),180$     ;
                                0318  1505            SETBIT  LPD$V_RBF,LPD$W_STS(R6) ; Mark for buffered receives
 00000000'8F    E1              0320  1506 180$:     BBC     #IO$_WRITELBLK,-        ; If BC then direct I/O function
                04 08 A0        0326  1507                    FDT_IOTYPE(R0),200$     ;
                                0329  1508            SETBIT  LPD$V_XBF,LPD$W_STS(R6) ; Mark for buffered transmissions
                                032D  1509 200$:     ;
                                032D  1510            ;   If this is a broadcast circuit, then do not allow the circuit
                                032D  1511            ;   TRANSPORT TYPE to be any value other than Phase IV values.
                                032D  1512            ;
       24 22 A6    0A   E1     032D  1513            BBC     #LPD$V_BC,LPD$W_STS(R6),205$ ; If broadcast circuit,
                                0332  1514            $GETFLD cri,l_xpt               ; Get transport type parameter
             14 50      E9     033F  1515            BLBC    R0,205$                 ; Skip if not specified
              1837      30     0342  1516            BSBW    XPT_TO_PTY              ; Translate XPT to node type
                                0345  1517            $DISPATCH R0,<=                 ; Allow only if set to one of:
                                0345  1518                    <ADJ$C_PTY_PH4,205$>,-  ; Phase IV routing
                                0345  1519                    <ADJ$C_PTY_PH4N,205$>,- ; Phase IV endnode
                                0345  1520                    <ADJ$C_PTY_AREA,205$>>  ; Phase IV area routing
        50  0000'8F    3C     034F  1521            MOVZWL  #SS$_BADPARAM,R0        ; Else, illegal protocol
              58      11      0354  1522            BRB     300$                    ; Exit with error
                                0356  1523 205$:     ;
                                0356  1524            ;   Allocate an ADJ control block, to maintain adjacency-related
                                0356  1525            ;   parameters.  The first MAX_LPD slots in the ADJ vector correspond
                                0356  1526            ;   exactly to the LPD of the same index, so allocation is easy.
                                0356  1527            ;
          57    20 A6    9A     0356  1528            MOVZBL  LPD$B_PTH_INX(R6),R7    ; Get LPD index
        57  2C B447    D0      035A  1529            MOVL    @RCB$C_PTR_ADJ(R4)[R7],R7 ; Get pointer to ADJ block
              54      DD        035F  1530            PUSHL   R4                      ; Save registers
67  0D  00  6E    00   2C      0361  1531            MOVC5   #0,(SP),#0,#ADJ$C_LENGTH,(R7) ; Zero ADJ cell
              54  8ED0        0367  1532            POPL    R4                      ; Restore registers
        02 A7    20 A6    B0     036A  1533            MOVW    LPD$W_PTH(R6),ADJ$W_LPD(R7) ; Store associated LPD index in ADJ
        01 A7    FF 8F    90     036F  1534            MOVB    #ADJ$C_PTY_UNK,ADJ$B_PTYPE(R7) ; Mark partner type unknown
                                0374  1535            SETBIT  ADJ$V_INUSE,ADJ$B_STS(R7)   ; Mark cell in use
                                0377  1536            ;
                                0377  1537            ;   For broadcast circuits, initialize the "designated router"
                                0377  1538            ;   to be the NI itself.  For point-to-point circuits, initialize
                                0377  1539            ;   the "designated router" to be the remote partner node.
                                0377  1540            ;
             20 A6    9B        0377  1541            MOVZBW  LPD$B_PTH_INX(R6),-     ; Preset ADJ index of "DRT" to that of
             2C A6              037A  1542                    LPD$W_DRT(R6)           ; the circuit itself ("none")
                                037C  1543            ;
                                037C  1544            ;   If this is a broadcast circuit, then allocate a buffer to hold
                                037C  1545            ;   the current state of the "router election".
                                037C  1546            ;
                                037C  1547            $DISPATCH RCB$B_ETY(R4),TYPE=B,<- ; If we are an endnode,
                                037C  1548                    <ADJ$C_PTY_PH4N,210$>,- ; skip the following
                                037C  1549                    <ADJ$C_PTY_PH3N,210$>>
    12 22 A6    0A   E1        038C  1550            BBC     #LPD$V_BC,LPD$W_STS(R6),210$ ; Skip if non-broadcast circuit
51    000000F9  8F   D0        0391  1551            MOVL    #12+1+TR4C_MAX_RSLIST,R1 ; Set size of buffer needed
             FC65'  30          0398  1552            BSBW    NET$ALONPGD_Z           ; Allocate buffer from nonpaged pool
              10 50    E9       039B  1553            BLBC    R0,300$                 ; Branch if error detected
    2E A6    0C A2    9E       039E  1554            MOVAB   12(R2),LPD$L_RTR_LIST(R6) ; Store address of buffer
                                03A3  1555 210$:     ;
                                03A3  1556            ;   Link the new LPD to the CNF block
                                03A3  1557            ;
        58    20 A6    3C       03A3  1558            MOVZWL  LPD$W_PTH(R6),R8        ; Get path i.d.
        12 AA    58    B0       03A7  1559            MOVW    R8,CNF$W_ID(R10)        ; Link CNF to LPD
           50    00'   D0       03AB  1560            MOVL    S^#SS$_NORMAL,R0        ; Indicate success
```

```
                OF 50    E8  03AE  1561 300$:   BLBS    R0,390$                 ; If error on exit,
                   56    D5  03B1  1562          TSTL    R6                      ; Was an LPD allocated?
                   0B    13  03B3  1563          BEQL    390$                    ; Branch if not
              0201 8F    BB  03B5  1564          PUSHR   #^M<R0,R9>              ; Save final status
                 0091    30  03B9  1565          BSBW    DEAL_LPD                ; If so, cleanup LPD
              0201 8F    BA  03BC  1566          POPR    #^M<R0,R9>              ; Restore final status
                         05  03C0  1567 390$:    RSB                             ; Return status in R0
```

```
                                03C1  1569          .SBTTL  ALLOC_COSTHOPS - Allocate a cost/hops buffer
                                03C1  1570 ;+
                                03C1  1571 ; ALLOC_COSTHOPS - Allocate a cost/hops buffer for an adjacency
                                03C1  1572 ;
                                03C1  1573 ; This routine is called to allocate a buffer in ACP process space
                                03C1  1574 ; to hold the last routing message received from the circuit or
                                03C1  1575 ; broadcast router.  This is called when a circuit LPD is created,
                                03C1  1576 ; or when we hear from a new broadcast router.  The address of
                                03C1  1577 ; the buffer is stored in the NET$AL_CH_VEC pointer vector.
                                03C1  1578 ;
                                03C1  1579 ; Inputs:
                                03C1  1580 ;
                                03C1  1581 ;       R8 = ADJ index
                                03C1  1582 ;
                                03C1  1583 ; Outputs:
                                03C1  1584 ;
                                03C1  1585 ;       R0 = status code
                                03C1  1586 ;
                                03C1  1587 ;       R1-R2 is destroyed.
                                03C1  1588 ;-
                                03C1  1589 ALLOC_COSTHOPS:
                      38    BB  03C1  1590          PUSHR   #^M<R3,R4,R5>           ; Save registers
          00000000'EF       16  03C3  1591          JSB     NET$GET_RTG2            ; Get routing info
                   55 50    E9  03C9  1592          BLBC    R0,90$                  ; Branch if error
     51    0000080C 8F       D0  03CC  1593          MOVL    #12+<2*NUM_NODES>,R1    ; Set length of block
                 FC2A'       30  03D3  1594          BSBW    NET$ALLOCATE            ; Allocate the buffer
                   48 50    E9  03D6  1595          BLBC    R0,90$                  ; Branch if unable to allocate
                52    0C    C0  03D9  1596          ADDL    #12,R2                  ; Point to first available byte
                51    0C    C2  03DC  1597          SUBL    #12,R1
     00000980'EF48    52    D0  03DF  1598          MOVL    R2,NET$AL_CH_VEC[R8]    ; Store address in vector
62   51    FF 8F    6E    00  2C  03E7  1599          MOVC5   #0,(SP),#-1,R1,(R2)     ; Initialize it to max cost/hops
                                03EE  1600          ;
                                03EE  1601          ;   If we are an area router, then allocate one for area
                                03EE  1602          ;   routing messages as well.
                                03EE  1603          ;
          51    00000000'EF       D0  03EE  1604          MOVL    NET$GL_PTR_VCB,R1       ; Get RCB address
             03    008A C1       91  03F5  1605          CMPB    RCB$B_ETY(R1),#ADJ$C_PTY_AREA ; Are we an area router?
                          22       12  03FA  1606          BNEQ    80$                     ; If not, then exit
          51    0000008C 8F       D0  03FC  1607          MOVL    #12+<2*NUM_AREAS>,R1    ; Set length of block
                        FBFA'       30  0403  1608          BSBW    NET$ALLOCATE            ; Allocate the buffer
                           18 50    E9  0406  1609          BLBC    R0,90$                  ; Branch if unable to allocate
                         52    0C    C0  0409  1610          ADDL    #12,R2                  ; Point to first available byte
                         51    0C    C2  040C  1611          SUBL    #12,R1
     00001A88'EF48    52    D0  040F  1612          MOVL    R2,NET$AL_AREA_CH[R8]   ; Store address in vector
62   51    FF 8F    6E    00  2C  0417  1613          MOVC5   #0,(SP),#-1,R1,(R2)     ; Initialize it to max cost/hops
                         50    01    D0  041E  1614 80$:     MOVL    #1,R0                   ; Success
                                38    BA  0421  1615 90$:     POPR    #^M<R3,R4,R5>           ; Restore registers
                                05    05  0423  1616          RSB
```

```
                                  0424          .SBTTL  DEAL_LPD - Deallocate LPD
                                  0424  ;++
                                  0424  ; COND_DEAL_LPD - Conditionally deallocate LPD
                                  0424  ; DEAL_LPD      - Unconditionally deallocate LPD
                                  0424  ;
                                  0424  ; The I/O channel is $DEASSGN'd, and the LPD block is deallocated.
                                  0424  ; The LPD is unhooked from the CRI CNF.
                                  0424  ;
                                  0424  ; INPUTS:       R11     CRI CNR pointer
                                  0424  ;               R10     CRI CNF pointer
                                  0424  ;               R6      LPD pointer
                                  0424  ;
                                  0424  ; OUPUTS:       R6      Zero
                                  0424  ;               R0      LBS if successful
                                  0424  ;                       LBC otherwise
                                  0424  ;
                                  0424  ;               R1-R4,R7-R9 are destroyed
                                  0424  ;--
                                  0424  COND_DEAL_LPD:                          ; Conditionally deallocate LPD
                                  0424          $GETFLD cri,l,sta               ; Get the operater state
                 07 50     E9     0431          BLBC    R0,10$                  ; If LBC then assume "off"
                    50     D4     0434          CLRL    R0                      ; Assume can't deallocate
              01    58     91     0436          CMPB    R8,#NMA$C_STATE_OFF     ; Is the state "off"
                    11     12     0439          BNEQ    20$                     ; If NEQ then can't deallocate
                 1B A6     95     043B  10$:    TSTB    LPD$B_ASTCNT(R6)        ; Has LPD run-down?
                    0C     12     043E          BNEQ    20$                     ; If NEQ no, return error
                 1C A6     95     0440          TSTB    LPD$B_IRPCNT(R6)        ; Does NETDRIVER still have references?
                    07     12     0443          BNEQ    20$                     ; If NEQ, then wait for NETDRIVER
                                  0445                                          ; to wake us up with CRD event
                    03     E0     0445          BBS     #LPD$V_ACCESS,-         ; If accessed for "service" then
              02 22 A6            0447                      LPD$W_STS(R6),20$   ; cannot deallocate
                    01     10     044A          BSBB    DEAL_LPD                ; Deallocate LPD
                           05     044C  20$:    RSB                             ; Done
                                  044D
                                  044D  DEAL_LPD:                               ; Deallocate LPD
           50    28 A6     9A     044D          MOVZBL  LPD$B_PLVEC(R6),R0      ; Get PLVEC index
                    15     13     0451          BEQL    10$                     ; If EQL then none
                 28 A6     94     0453          CLRB    LPD$B_PLVEC(R6)         ; Init the PLVEC index
        00000000'EF40      97     0456          DECB    PLVEC$AB_REFC[R0]       ; No longer referencing it
  14 A6 00000000'EF40      B1     045D          CMPW    PLVEC$AW_CHAN[R0],-     ; Are the line and circuit channels
                                  0466                  LPD$W_CHAN(R6)          ; the same?
                    0B     13     0466          BEQL    15$                     ; If so, let line-related code deassign it
                                  0468  10$:    $DASSGN_S CHAN = LPD$W_CHAN(R6) ; De-assign channel
                 12 AA     B4     0473  15$:    CLRW    CNF$W_ID(R10)           ; Unbind LPD from CRI
           54    20 A6     3C     0476          MOVZWL  LPD$W_PTH(R6),R4        ; Get current path index & seq. no
              52    54     9A     047A          MOVZBL  R4,R2                   ; Get LPD index
   50   00000980'EF42      D0     047D          MOVL    NET$AL_CH_VEC[R2],R0    ; Get address of routing msg buffer
                    0D     13     0485          BEQL    20$                     ; Branch if none
              50    0C     C2     0487          SUBL    #12,R0                  ; Point to real start of block
                 FB73'     30     048A          BSBW    NET$DEALLOCATE          ; Deallocate routing message buffer
        00000980'EF42      D4     048D          CLRL    NET$AL_CH_VEC[R2]       ; Invalidate pointer
   50   00001A88'EF42      D0     0494  20$:    MOVL    NET$AL_AREA_CH[R2],R0   ; Get address of area routing buffer
                    0D     13     049C          BEQL    25$                     ; Branch if none
              50    0C     C2     049E          SUBL    #12,R0                  ; Point to real start of block
                 FB5C'     30     04A1          BSBW    NET$DEALLOCATE          ; Deallocate routing message buffer
        00001A88'EF42      D4     04A4          CLRL    NET$AL_AREA_CH[R2]      ; Invalidate pointer
     51   00000000'EF      D0     04AB  25$:    MOVL    NET$GL_PTR_VCB,R1       ; Get RCB address
```

```
          50   2C B142   D0  04B2  1675          MOVL    @RCB$L_PTR_ADJ(R1)[R2],R0 ; Get address of ADJ block
                    16   BB  04B7  1676          PUSHR   #^M<R1,R2,R4>             ; Save registers
 60   0D   00   6E   00   2C  04B9  1677          MOVC5   #0,(SP),#0,#ADJ$C_LENGTH,(R0) ; Zero ADJ - including INUSE bit
                    16   BA  04BF  1678          POPR    #^M<R1,R2,R4>             ; Restore registers
            50   2E A6   D0  04C1  1679          MOVL    LPD$L_RTR_LIST(R6),R0     ; Get address of RTR_LIST buffer
                    06   13  04C5  1680          BEQL    30$                       ; Branch if none
               50   0C   C2  04C7  1681          SUBL    #12,R0                    ; Point to real start of block
                  FB33'  30  04CA  1682          BSBW    NET$DEALLOCATE            ; Deallocate the buffer
               50   56   D0  04CD  1683 30$:     MOVL    R6,R0                     ; Point to LPD structure
                  FB2D'  30  04D0  1684          BSBW    NET$DEALLOCATE            ; Deallocate LPD block
                    56   D4  04D3  1685          CLRL    R6                        ; Invalidate LPD pointer
          28 B142   54   D0  04D5  1686          MOVL    R4,@RCB$L_PTR_LPD(R1)[R2] ; Invalidate LPD vector slot
                              04DA  1687                                            ; and store current index & seq. no
                              04DA  1688                                            ; instead of a pointer (bit 31 clear)
               50   00'   D0  04DA  1689          MOVL    S^#SS$_NORMAL,R0          ; Setup status
                          05  04DD  1690          RSB
```

```
                                                     J 5
                     04DE  1692          .SBTTL  CHECK_REQ_PARAMS - Check that required parameters are set
                     04DE  1693 ;+
                     04DE  1694 ; CHECK_REQ_PARAMS - Check that required circuit parameters are specified
                     04DE  1695 ;
                     04DE  1696 ; This routine is called when a circuit is turned on, in order to ensure
                     04DE  1697 ; that the proper parameters were specified, depending on the type of
                     04DE  1698 ; circuit.  This is done here, so that immediate feedback can be given
                     04DE  1699 ; to the requestor.
                     04DE  1700 ;
                     04DE  1701 ; Inputs:
                     04DE  1702 ;
                     04DE  1703 ;       R11 = CRI CNR address
                     04DE  1704 ;       R10 = CRI CNF address
                     04DE  1705 ;
                     04DE  1706 ; Outputs:
                     04DE  1707 ;
                     04DE  1708 ;       R0 = status code
                     04DE  1709 ;-
                     04DE  1710 CHECK_REQ_PARAMS:
                     04DE  1711          ;
                     04DE  1712          ;   COST must be specified for all routing circuits.
                     04DE  1713          ;
                     04DE  1714          $GETFLD cri,l,cos                   ; Get the COST value
          03 50   E8 04EB  1715          BLBS    R0,10$                      ; Branch if okay
           0080   31 04EE  1716          BRW     80$                         ; Else, error
                     04F1  1717          ;
                     04F1  1718          ;   If we are an endnode, do not allow TRANSPORT TYPE
                     04F1  1719          ;   parameter to be set to a router.
                     04F1  1720          ;
    54 00000000'EF   D0 04F1  1721 10$:    MOVL    NET$GL_PTR_VCB,R4           ; Get RCB address
                     04F8  1722          $DISPATCH RCB$B_ETY(R4),TYPE=B,<-   ; If we are an endnode,
                     04F8  1723                  <ADJ$C_PTY_PH4N,15$>,-
                     04F8  1724                  <ADJ$C_PTY_PH3N,15$>>
              28   11 0508  1725          BRB     20$
                     050A  1726 15$:     $GETFLD cri,l,xpt                   ; Get TRANSPORT TYPE
            18 50   E9 0517  1727          BLBC    R0,20$                      ; If specified,
             165F   30 051A  1728          BSBW    XPT_TO_PTY                  ; Translate XPT to node type
                     051D  1729          $DISPATCH R8,<=                     ; These are the allowable values
                     051D  1730                  <ADJ$C_PTY_PH4N,20$>,-
                     051D  1731                  <ADJ$C_PTY_PH3N,20$>>
    50 0000'8F   3C 052B  1732          MOVZWL  #SS$_BADPARAM,R0            ; Illegal parameter
              44   11 0530  1733          BRB     90$
                     0532  1734 20$:     ;
                     0532  1735          ;   If X.25 circuit, then check additional parameters
                     0532  1736          ;
                     0532  1737          $GETFLD cri,l,typ                   ; Get circuit type
          2A 50   E9 053F  1738          BLBC    R0,50$                      ; Branch if not set
          03 58   D1 0542  1739          CMPL    R8,#NMA$C_CIRTY_X25         ; X.25 circuit?
              25   12 0545  1740          BNEQ    50$                         ; Branch if not
                     0547  1741          ;
                     0547  1742          ;   For X.25 circuits, USAGE must be specified, and for outgoing
                     0547  1743          ;   DLM circuits, NUMBER must be specified.
                     0547  1744          ;
                     0547  1745          $GETFLD cri,l,use                   ; Get USAGE
          1A 50   E9 0554  1746          BLBC    R0,80$                      ; Error if not specified
          02 58   D1 0557  1747          CMPL    R8,#NMA$C_CIRUS_OUT         ; Outgoing?
              10   12 055A  1748          BNEQ    50$                         ; Branch if not
```

```
                        055C  1749          $GETFLD cri,l,num                ; Get NUMBER
          05 50    E9   0569  1750          BLBC    R0,80$                   ; Error if not specified
        50   00'   D0   056C  1751 50$:     MOVL    S^#SS$_NORMAL,R0         ; Successful
             05    11   056F  1752          BRB     90$
   50   0000'8F    3C   0571  1753 80$:     MOVZWL  #SS$_INSFARG,R0          ; COST value missing
                  05   0576  1754 90$:     RSB
```

```
                                0577  1756          .SBTTL  NET$DLL_X25_CALL - Process incoming X.25 call
                                0577  1757 ;+
                                0577  1758 ; NET$DLL_X25_CALL - Process incoming X.25 call
                                0577  1759 ;
                                0577  1760 ; Attempt to associate the incoming call with a waiting X.25 DLM circuit
                                0577  1761 ; which is marked "waiting for incoming call".  If a circuit is found,
                                0577  1762 ; queue an event to the circuit.
                                0577  1763 ;
                                0577  1764 ; Inputs:
                                0577  1765 ;
                                0577  1766 ;       R9 = Unit number reported in mailbox message
                                0577  1767 ;       R10/R11 = Descriptor of message data in mailbox message
                                0577  1768 ;                 (which is a byte-counted string containing incoming NCB)
                                0577  1769 ;
                                0577  1770 ; Outputs:
                                0577  1771 ;
                                0577  1772 ;       None
                                0577  1773 ;-
                                0577  1774 NET$DLL_X25_CALL::
            5A    8B    9A    0577  1775          MOVZBL  (R11)+,R10              ; Construct descriptor of incoming NCB
            7E    5A    7D    057A  1776          MOVQ    R10,-(SP)              ; Save NCB descriptor on stack
                  7E    7C    057D  1777          CLRQ    -(SP)                  ; Preset descriptor of remote DTE
                              057F  1778 ;
                              057F  1779 ;   Locate the remote DTE address in the NCB
                              057F  1780 ;
         01     02 AB    B1    057F  1781 10$:    CMPW    2(R11),#PSI$C_NCB_REMDTE ; Have we found remote DTE entry?
                  09    12    0583  1782          BNEQ    15$                    ; If not, continue looking
            6E    04 AB    9A    0585  1783          MOVZBL  4(R11),(SP)            ; Save descriptor of remote DTE string
      04    AE    05 AB    9E    0589  1784          MOVAB   5(R11),4(SP)
            50    6B    3C    058E  1785 15$:    MOVZWL  (R11),R0               ; Get length of entry
            5B    50    C0    0591  1786          ADDL    R0,R11                 ; Skip to next entry
            5A    50    C2    0594  1787          SUBL    R0,R10                 ; Subtract from length of NCB left
                  E6    14    0597  1788          BGTR    10$                    ; If more left, continue search
                              0599  1789 ;
                              0599  1790 ;   Search for an incoming circuit, waiting for a call,
                              0599  1791 ;   and which matches the remote DTE address, if the
                              0599  1792 ;   incoming circuit was restricted to a given remote DTE.
                              0599  1793 ;
         5B   00000000'EF     D0    0599  1794          MOVL    NET$GL_CNR_CRI,R11     ; Get address of CRI root
                        5A    D4    05A0  1795          CLRL    R10                    ; Start at beginning of list
                     58   01    D0    05A2  1796 30$:    MOVL    #NMA$C_CIRUS_INC,R8    ; Set value of "incoming"
                                  05A5  1797          $SEARCH eql,cri,l,use          ; Search for USAGE INCOMING circuits
                       4E 50    E9    05B4  1798          BLBC    R0,50$                 ; Reject call if none found
                         26E2   30    05B7  1799          BSBW    NET$LOCATE_LPD         ; Locate LPD associated with circuit
                       E5 50    E9    05BA  1800          BLBC    R0,30$                 ; If none, ignore circuit
                                  05BD  1801          $GETFLD cri,s,num              ; Get specific remote DTE, if specified
                       09 50    E9    05CA  1802          BLBC    R0,35$                 ; If not specified, allow everybody
04    BE    6E    00   68    57    2D    05CD  1803          CMPC5   R7,(R8),#0,(SP),a4(SP) ; Does the remote DTE match?
                     CC   12    05D4  1804          BNEQ    30$                    ; If not, skip this circuit
                  C7 22 A6    09    E5    05D6  1805 35$:    BBCC    #LPD$V_INCOMING,LPD$W_ST$(R6),30$ ; Check if waiting for call
                                  05DB  1806                                         ; and mark it "no longer waiting"
                                  05DB  1807 ;
                                  05DB  1808 ; Circuit found - queue event to circuit with WQE containing
                                  05DB  1809 ; the actual X.25 NCB for the incoming call.
                                  05DB  1810 ;
                     5E   08    C0    05DB  1811          ADDL    #8,SP                  ; Pop remote DTE descriptor off stack
                     5A   8E    7D    05DE  1812          MOVQ    (SP)+,R10              ; Retreive NCB descriptor
```

```
                51   5A    D0  05E1  1813          MOVL    R10,R1                          ; Set size of extra WQE space
                50   01    D0  05E4  1814          MOVL    #WQE$C_SUB_ACP,R0               ; Set type of WQE
                   FA16'    30  05E7  1815          BSBW    WQE$ALLOCATE                    ; Allocate a WQE
             10 A2   17    90  05EA  1816          MOVB    #LEV$C_X25_CALL,WQE$B_EVT(R2)   ; Set event code
          12 A2   20 A6    B0  05EE  1817          MOVW    LPD$W_PTH(R6),WQE$W_REQIDT(R2)  ; Set path ID
             14 A2   5A    D0  05F3  1818          MOVL    R10,WQE$L_PM2(R2)               ; Set size of NCB
                   52    DD  05F7  1819          PUSHL   R2                              ; Save WQE address
       24 A2   6B   5A    28  05F9  1820          MOVC    R10,(R11),WQE$C_LENGTH(R2) ; Copy incoming NCB into WQE
                   55 8ED0  05FE  1821          POPL    R5                              ; Restore WQE address into R5
                  077C    30  0601  1822          BSBW    NET$DLL_PRC_WQE                 ; Process event and deallocate WQE
                          05  0604  1823          RSB
                              0605  1824
                              0605  1825 ;
                              0605  1826 ; No circuit could be found to handle the call.  Issue a QIO to reject it.
                              0605  1827 ;
                5E   08   C0  0605  1828 50$:    ADDL    #8,SP                           ; Pop remote DTE descriptor off stack
                50   5E   D0  0608  1829          MOVL    SP,R0                           ; Make pointer to NCB descriptor
                              060B  1830          $QIO_S  CHAN=NET$GW_X25_CHAN,-          ; Reject incoming call
                              060B  1831                  FUNC=#IO$_ACCESS!IO$M_ABORT,-
                              060B  1832                  P2=R0
                5E   08   C0  062C  1833          ADDL    #8,SP                           ; Pop NCB descriptor
                          05  062F  1834          RSB                                     ; and exit
```

```
                                                N 5
                     0630  1836          .SBTTL  NET$DLL_X25_RESET - X.25 reset detected
                     0630  1837 ;+
                     0630  1838 ; NET$DLL_X25_RESET - X.25 circuit was reset by other side
                     0630  1839 ;
                     0630  1840 ; This routine is called when a mailbox message is received from PSI
                     0630  1841 ; indicating that the X.25 circuit has been reset.  Our action is to
                     0630  1842 ; issue a "reset confirmation", allowing the reset operation on the
                     0630  1843 ; other side to complete.  This is primarily needed during datalink
                     0630  1844 ; initialization, when there is no receive IRP available to detect
                     0630  1845 ; reset requests on the circuit.
                     0630  1846 ;
                     0630  1847 ; Inputs:
                     0630  1848 ;
                     0630  1849 ;       R9 = Unit number reported in mailbox message
                     0630  1850 ;       R10/R11 = Descriptor of message data in mailbox message
                     0630  1851 ;                 (which is 3 bytes of: diagnostic, cause, reason)
                     0630  1852 ;
                     0630  1853 ; Outputs:
                     0630  1854 ;
                     0630  1855 ;       None
                     0630  1856 ;-
                     0630  1857 NET$DLL_X25_RESET::
                     0630  1858                 ;
                     0630  1859                 ;   Find the LPD whose channel corresponds to the unit
                     0630  1860                 ;   number in the mailbox message.
                     0630  1861                 ;
54   00000000'EF  D0 0630  1862                 MOVL    NET$GL_PTR_VCB,R4       ; Get RCB address
     55   5C A4   9A 0637  1863                 MOVZBL  RCB$B_MAX_LPD(R4),R5    ; Get number of LPDs
            1B    13 063B  1864                 BEQL    30$                     ; If none, ignore message
  56   28 B445  D0 063D  1865 10$:    MOVL    @RCB$L_PTR_LPD(R4)[R5],R6 ; Get LPD address
            11    18 0642  1866                 BGEQ    20$                     ; Branch if slot not valid
0C 22 A6    07  E1 0644  1867                 BBC     #LPD$V_X25,LPD$W_STS(R6),20$ ; Skip if not X.25 circuit
  50   10 A6   D0 0649  1868                 MOVL    LPD$L_UCB(R6),R0        ; Get UCB address
            06    13 064D  1869                 BEQL    20$                     ; Skip if no datalink
54 A0       59  B1 064F  1870                 CMPW    R9,UCB$W_UNIT(R0)       ; Does unit number match?
            04    13 0653  1871                 BEQL    50$                     ; Exit loop if it matches
     E5 55      F5 0655  1872 20$:    SOBGTR  R5,10$                  ; Loop through all LPDs
               05  0658  1873 30$:    RSB                             ; Ignore mailbox message
                   0659  1874                 ;
                   0659  1875                 ;   We have found the proper X.25 circuit.  Queue an event.
                   0659  1876                 ;
        51         D4 0659  1877 50$:    CLRL    R1                      ; No extra WQE space needed
   50   01         D0 065B  1878                 MOVL    #WQE$C_SUB_ACP,R0       ; Indicate type of WQE
     F99F'         30 065E  1879                 BSBW    WQE$ALLOCATE            ; Allocate a work queue entry
   55   52         D0 0661  1880                 MOVL    R2,R5                   ; Copy WQE address
10 A5   19         90 0664  1881                 MOVB    #LEV$C_X25_RESET,WQE$B_EVT(R5) ; Set event code
12 A5   20 A6      B0 0668  1882                 MOVW    LPD$W_PTH(R6),WQE$W_REQIDT(R5) ; Set path ID
      0710         30 066D  1883                 BSBW    NET$DCL_PRC_WQE         ; Process event and deallocate WQE
                   05 0670  1884                 RSB
```

```
0671  1886          .SBTTL  NET$DLL_RCV - Process message received from driver
0671  1887 ;+
0671  1888 ; NET$DLL_RCV - Process block received from the Transport layer
0671  1889 ;
0671  1890 ; FUNCTIONAL DESCRIPTION:
0671  1891 ;
0671  1892 ; Received messages are passed to the ACP from NETDRIVER by queuing the non-
0671  1893 ; paged DYN$C_NET buffer directly to the ACP's AQB.  The WQE header and the
0671  1894 ; body of the message are stored within the same buffer.  The message is
0671  1895 ; scanned to determine its type, an event code is generated, and the event is
0671  1896 ; dispatched.
0671  1897 ;
0671  1898 ; When a datalink is initialized, NETDRIVER allocates a single IRP for queuing
0671  1899 ; receives to the datalink.  Post processing for this IRP takes place in
0671  1900 ; NETDRIVER which detaches the received buffer and recycles the IRP by queuing
0671  1901 ; it again to the same datalink.  However, prior to recycling the IRP, if the
0671  1902 ; XM$B_STS_ACTIVE bit in IRP$L_IOST2 is clear then NETDRIVER realizes that the
0671  1903 ; device has shutdown and passes the IRP to the ACP instead of the datalink.
0671  1904 ; The ACP comes here to process this returned IRP.  The eventual action should
0671  1905 ; be to read the entire IRP$L_IOST2 image to detect such things as device
0671  1906 ; entering maintenance mode and to log this event.  For now, the IRP is assumed
0671  1907 ; to be a signal that the device has shutdown.
0671  1908 ;
0671  1909 ; On return, the block is eventually deallocated.
0671  1910 ;
0671  1911 ; INPUTS:       R5      WQE ptr
0671  1912 ;
0671  1913 ;               All others are scratch.
0671  1914 ;
0671  1915 ; OUTPUTS:      All registers are clobbered.
0671  1916 ;
0671  1917 ;-
0671  1918
0671  1919 ;
0671  1920 ;  The expected messages have the following format:
0671  1921 ;
0671  1922 ;
0671  1923 ; Phase 2 init      <0101 1000><1K_1><EX2_add><I6_nam><1B_fct><1B_req>-
0671  1924 ;                                <2B_blksiz><2B_nspsiz><2B_maxlnk>-
0671  1925 ;                                <3b_rtver><3b_comver><I32_sysid>
0671  1926 ; Phase 2 verf      <0101 1000><1K_2><8B_psw>
0671  1927 ;
0671  1928 ;                       <1B_fct>    ::== <1k_0>          no intercept functions
0671  1929 ;                                        <1k_7>          intercept functions
0671  1930 ;
0671  1931 ;                       <1B_req>    ::== low bit = 0 => verf requested
0671  1932 ;                                        low bit = 1 => no verf requested
0671  1933 ;                                        ignore other requests
0671  1934 ;
0671  1935 ;                       <3B_rtver>  ::== <1K_3><1K_1><1K_0>
0671  1936 ;                       <3B_comver> ::== <1K_3><1K_1><1K_0>
0671  1937 ;
0671  1938 ;
0671  1939 ; Phase 3 init      <0000 0001><2B_srcnode><1B_tiinfo><2B_blksiz>-
0671  1940 ;                                        <3b_tiver><I64_seed>
0671  1941 ; Phase 3/4 verf    <0000 0011><2B_srcnode><I64_psw>
0671  1942 ; Phase 3/4 test    <0000 0101><2B_srcnode><I128_data>
```

```
                        0671  1943 ; Phase 3 rout      <0000 0111><2B_srcnode><rtginfo><checksum>
                        0671  1944 ;
                        0671  1945 ;                       <1B_tiinfo> ::== <0000bvnn>  nn = 00 reserved
                        0671  1946 ;                                                      = 01 reserved
                        0671  1947 ;                                                      = 10 routing
                        0671  1948 ;                                                      = 11 nonrouting
                        0671  1949 ;                                                    v =  0 no verf requested
                        0671  1950 ;                                                      =  1 verf requested
                        0671  1951 ;                                                    b =  0 no DLM blocking
                        0671  1952 ;                                                      =  1 DLM block requested
                        0671  1953 ;
                        0671  1954 ;                       <3B_tiver>  ::== <1K_1><1K_3><1K_0>
                        0671  1955 ;                       <64I_seed>  ::== <1K_0>
                        0671  1956 ;
                        0671  1957 ; Phase 4 init      <0000 0001><2B_srcnode><1B_tiinfo><2B_blksiz>-
                        0671  1958 ;                                          <3b_tiver><2B_hello><I64_seed>
                        0671  1959 ; Phase 4 rout      <0000 0111><2B_srcnode><1K_0><rtginfo><checksum>
                        0671  1960 ; Phase 4 area rout <0000 1001><2B_srcnode><1K_0><rtginfo><checksum>
                        0671  1961 ;
                        0671  1962 ;                       <1B_tiinfo> ::== <0000bvnn>  nn = 00 reserved
                        0671  1963 ;                                                      = 01 area routing
                        0671  1964 ;                                                      = 10 routing
                        0671  1965 ;                                                      = 11 nonrouting
                        0671  1966 ;                                                    v =  0 no verf requested
                        0671  1967 ;                                                      =  1 verf requested
                        0671  1968 ;                                                    b =  0 no DLM blocking
                        0671  1969 ;                                                      =  1 DLM block requested
                        0671  1970 ;
                        0671  1971 ;                       <3B_tiver>  ::== <1K_2><1K_0><1K_0>
                        0671  1972 ;                       <64I_seed>  ::== <1K_0>
                        0671  1973 ;
                        0671  1974 ;
                        0671  1975 NET$DLL_RCV::                            ; Process received message
                        0671  1976 ;
                        0671  1977 ;   Establish the context for the event
                        0671  1978 ;
00000034'EF    94       0671  1979         CLRB    XMTFLG                   ; Clear all xmit flags
00000038'EF    94       0677  1980         CLRB    PTYPE                    ; Clear partner node type
0000000C'EF    D4       067D  1981         CLRL    LEV_L_LPD                ; Clear the LPD pointer
00000010'EF    D4       0683  1982         CLRL    LEV_L_ADJ                ; Clear the ADJ pointer
00000004'EF    7C       0689  1983         CLRQ    LEV_Q_CRI                ; Clear the CRI CNF,CNR ptrs
00000014'EF    B4       068F  1984         CLRW    LEV_W_PNA                ; Clear partner's node address
00000018'EF    B4       0695  1985         CLRW    LEV_W_BLKSIZE            ; Clear partner's block size
0000001C'EF    94       069B  1986         CLRB    LEV_B_PRIORITY           ; Clear router priority
00000020'EF    B4       06A1  1987         CLRW    LEV_W_HELLO              ; Clear partner's hello timer
00000024'EF    7C       06A7  1988         CLRQ    LEV_Q_PSWDESC            ; Clear init password descriptor
00000000'EF    D4       06AD  1989         CLRL    NET$GL_INITVER           ; Clear recevied INIT message version
      51  14  A5    3C  06B3  1990         MOVZWL  WQE$L_PM2(R5),R1         ; Get offset to message
          51  55    C0  06B7  1991         ADDL    R5,R1                    ; Convert to pointer
      18  A5  51    D0  06BA  1992         MOVL    R1,WQE$L_EVL_PKT(R5)     ; Store ptr in case packet header
                        06BE  1993                                          ; is logged
                0789    30  06BE  1994         BSBW    FIND_WQE_CTX         ; Locate CNF, LPD, ADJ blocks
              34  50    E9  06C1  1995         BLBC    R0,20$               ; If LPD no longer exists, skip event
0000000C'EF     56    D0  06C4  1996         MOVL    R6,LEV_L_LPD             ; Save the LPD pointer in case
                        06CB  1997                                          ; DISPATCH fails (for code below)
54    00000000'EF       D0  06CB  1998         MOVL    NET$GL_PTR_VCB,R4        ; Get the RCB pointer
                  28    10  06D2  1999         BSBB    DISPATCH                 ; Dispatch to determine the event
```

```
          03 50   E9  06D4  2000          BLBC    R0,10$                  ; If cannot determine, skip event
           06B9   30  06D7  2001          BSBW    PROC_EVT                ; Process the event
                      06DA  2002 10$:     ;
                      06DA  2003          ;   If LPD's receiver is suspended waiting for a buffer then pass this
                      06DA  2004          ;   this buffer back to NETDRIVER.  Else, deallocate it.
                      06DA  2005          ;
          FB'AF   9F  06DA  2006          PUSHAB  B^30$                   ; Setup return address
56   0000000C'EF  D0  06DD  2007          MOVL    LEV_L_LPD,R6            ; Get the LPD
             12   13  06E4  2008          BEQL    20$                     ; If EQL then none
     50   32 A6   D0  06E6  2009          MOVL    LPD$L_RCV_IRP(R6),R0    ; Is there a waiting receive IRP?
             0C   13  06EA  2010          BEQL    20$                     ; If EQL then none
     2C A0   55   D0  06EC  2011          MOVL    R5,IRP$L_SVAPTE(R0)     ; Attach buffer to it
             55   D4  06F0  2012          CLRL    R5                      ; ...and erase our pointer to it
        50   0C   D0  06F2  2013          MOVL    S^#NETUPD$_REACT_RCV,R0 ; Fct code is "reactivate receiver"
           2634   31  06F5  2014          BRW     TELL_NETDRIVER          ; Give the buffer back to NETDRIVER
           068D   30  06F8  2015 20$:     BSBW    KILL_WQE                ; Else, deallocate the buffer
                  05  06FB  2016 30$:     RSB
```

```
                        06FC  2018          .SBTTL  Received message pre-processing routines
                        06FC  2019 ;+
                        06FC  2020 ; These routines are called after receiving a message to pre-process (parse)
                        06FC  2021 ; the message, and store common results in known cells.  This partially masks
                        06FC  2022 ; the difference between various versions which are supported.
                        06FC  2023 ;
                        06FC  2024 ; Inputs:
                        06FC  2025 ;
                        06FC  2026 ;       R11     CNR address
                        06FC  2027 ;       R10     CNF address
                        06FC  2028 ;       R7      ADJ address
                        06FC  2029 ;       R6      LPD address
                        06FC  2030 ;       R5      WQE address
                        06FC  2031 ;       R4      RCB address
                        06FC  2032 ;
                        06FC  2033 ; Outputs:
                        06FC  2034 ;
                        06FC  2035 ;       WQE$B_EVT = Event to be queued to state transition mechanism.
                        06FC  2036 ;
                        06FC  2037 ;       All input registers must be preserved by the parsing routines.
                        06FC  2038 ;-
                        06FC  2039 DISPATCH:
          53   10 A5   9A 06FC  2040          MOVZBL  WQE$B_EVT(R5),R3        ; Get Transport layer event code
                        0700  2041          $DISPATCH R3,<-
                        0700  2042                  <NETMSG$C_IRP,   IRP>,- ; IRP event
                        0700  2043                  <NETMSG$C_UNK,   UNK>,- ; Possibly transport control message
                        0700  2044                  <NETMSG$C_APL,   APL>,- ; Aged packet
                        0700  2045                  <NETMSG$C_OPL,   OPL>,- ; Oversized packet loss
                        0700  2046                  <NETMSG$C_NOL,   NOL>,- ; Packet for out-of-range node
                        0700  2047                  <NETMSG$C_NUL,   NUL>,- ; Packet for unreachable node
                        0700  2048                  <NETMSG$C_PFE,   PFE>,- ; Packet with format error
                        0700  2049                  <NETMSG$C_LSN,   LSN>,- ; Listener timeout
                        0700  2050                  <NETMSG$C_CRD,   CRD>,- ; Circuit run down
                        0700  2051                  <NETMSG$C_ADJ,   ADJ>,- ; Adjacency up
                        0700  2052          >
                        071C  2053          BUG_CHECK  NETNOSTATE,FATAL     ; Bug if unknown
                        0720  2054
                        0720  2055 ;
                        0720  2056 ; The CRD message says that NETDRIVER has just completed it's last reference
                        0720  2057 ; to the LPD, so that it can be deallocated.  This is handled by queueing an
                        0720  2058 ; IRP_DOWN event, which causes the state table to eventually try and deallocate
                        0720  2059 ; the LPD again - which this time, will succeed.
                        0720  2060 ;
                        0720  2061 ; For the last IRP that NETDRIVER converts into a CRD message, the IRPCNT
                        0720  2062 ; in the LPD is not decremented until the message actually is processed by
                        0720  2063 ; NETACP.   This prevents any activity on the LPD until all relevant messages
                        0720  2064 ; have been handled.
                        0720  2065 ;
            1C A6   97  0720  2066 CRD:     DECB    LPD$B_IRPCNT(R6)        ; Indicate receipt of CRD message
                        0723  2067                                          ; (allow startup activity to continue)
             F8DA'  30  0723  2068          BSBW    NET$JNX_CO              ; Initialize journalling co-routine
          0D 50     E9  0726  2069          BLBC    R0,30$                  ; Branch if journalling not enabled
        81    33    90  0729  2070          MOVB    #^X33,(R1)+             ; Journal record type = Returned IRP
    81  20 A6    90  072C  2071          MOVB    LPD$B_PTH_INX(R6),(R1)+ ; LPD index
              81    B4  0730  2072          CLRW    (R1)+                   ; Indicate no I/O function code
              81    7C  0732  2073          CLRQ    (R1)+                   ; Indicate no I/O completion status
              9E    16  0734  2074          JSB     @(SP)+                  ; Log the journalling record
```

```
                10 A5   20   90   0736   2075 30$:    MOVB    #LEV$C_IRP_DOWN,WQE$B_EVT(R5) ; Device has shut down
                   50   01   D0   0739   2076         MOVL    #1,R0                         ; Process the event
                             05   073C   2077         RSB                                   ; Enter state transition
                                  073D   2078
                                  073D   2079
                                  073D   2080 ;
                                  073D   2081 ; IRP - "FATAL DATALINK I/O ERROR"
                                  073D   2082 ;
                                  073D   2083 ; Inputs:
                                  073D   2084 ;
                                  073D   2085 ;       WQE$W_REQIDT = LPD index
                                  073D   2086 ;
                                  073D   2087 ; An IRP was just returned from the datalink layer.  Check to see if it
                                  073D   2088 ; applies to the local LPD, because if so, it is a signal that NETDRIVER
                                  073D   2089 ; is shutting down.
                                  073D   2090 ;
01              12 A5   91   073D   2091 IRP:    CMPB    WQE$W_REQIDT(R5),#LPD$C_LOC_INX ; Local LPD index?
                   06   12   0741   2092         BNEQ    10$                             ; Branch if not
                 F8B9'  30   0743   2093         BSBW    NET$LOCLPD_DOWN                 ; Report NETDRIVER shutting down
                   50   D4   0746   2094         CLRL    R0                              ; Do not process any event
                             05   0748   2095         RSB
                                  0749   2096 ;
                                  0749   2097 ; An IRP was just returned for a standard LPD.  This is either due
                                  0749   2098 ; to the line going down, or we just entered MOP mode.  Set the appropriate
                                  0749   2099 ; event so we can enter the state table.
                                  0749   2100 ;
                1B A6   97   0749   2101 10$:    DECB    LPD$B_ASTCNT(R6)                ; Reduce NETACP's claim on the LPD
                                  074C   2102                                            ; (for it's receive IRP)
                 F8B0'  30   074C   2103         BSBW    NET$JNX_CO                      ; Initialize journalling co-routine
                   11   50   E9   074F   2104         BLBC    R0,30$                          ; Branch if journalling not enabled
                      81   33   90   0752   2105         MOVB    #^X33,(R1)+                     ; Journal record type = Returned IRP
                   81   20 A6   90   0755   2106         MOVB    LPD$B_PTH_INX(R6),(R1)+         ; LPD index
                   81   20 A5   B0   0759   2107         MOVW    IRP$W_FUNC(R5),(R1)+            ; I/O function code
                   81   38 A5   7D   075D   2108         MOVQ    IRP$L_IOST1(R5),(R1)+           ; I/O completion status
                        9E   16   0761   2109         JSB     @(SP)+                          ; Log the journalling record
                10 A5   1F   90   0763   2110 30$:    MOVB    #LEV$C_IRP_RESET,WQE$B_EVT(R5) ; Assume X.25 circuit was reset
          0000'8F   38 A5   B1   0767   2111         CMPW    IRP$L_IOSTT(R5),#SS$_RESET ; Was X.25 circuit reset?
                        12   13   076D   2112         BEQL    50$                             ; Branch if yes
                10 A5   20   90   076F   2113         MOVB    #LEV$C_IRP_DOWN,WQE$B_EVT(R5) ; Assume device has shut down
             09 22 A6   07   E0   0773   2114         BBS     #LPD$V_X25,LPD$W_STS(R6),50$  ; Don't check MOP if X.25
                        13   E1   0778   2115         BBC     #XM$V_ERR_MAINT,-             ; Br if not MOP mode
                   04 3C A5        077A   2116                 IRP$L_IOST2(R5),50$           ;
                10 A5   21   90   077D   2117         MOVB    #LEV$C_IRP_MM,WQE$B_EVT(R5) ; Device entered MOP mode
                   50   01   D0   0781   2118 50$:    MOVL    #1,R0                           ; Process the event
                             05   0784   2119         RSB                                     ; Enter state transition
                                  0785   2120
                                  0785   2121
                                  0785   2122 ;
                                  0785   2123 ; ADJ - "ADJACENCY UP"
                                  0785   2124 ;
                                  0785   2125 ; Inputs:
                                  0785   2126 ;
                                  0785   2127 ;       WQE$W_REQIDT = LPD index
                                  0785   2128 ;       WQE$L_PM2 = Descriptor of message (word of length, word of offset)
                                  0785   2129 ;
                                  0785   2130 ; For adjacency up message, parse the message received by NETDRIVER,
                                  0785   2131 ; and if the message makes sense, then create an adjacency block
```

```
                     0786  2132 ; for the new Router or Endnode.  The only messages that are allowed
                     0786  2133 ; are: Start, Router Hello and Endnode Hello.  All other types of
                     0786  2134 ; messages are ignored.
                     0786  2135 ;
      0073    30     0786  2136 ADJ:    BSBW    UNK                             ; Parse received message
    3F 50     E9     0789  2137         BLBC    R0,40$                          ; If cannot parse, then ignore it
                     078C  2138         $DISPATCH WQE$B_EVT(R5),TYPE=B,<-       ; Based on type of message,
                     078C  2139                 <LEV$C_RCV_RHEL,10$>,-          ; Router Hello message
                     078C  2140                 <LEV$C_RCV_EHEL,20$>,-          ; Endnode Hello message
                     078C  2141                 <LEV$C_RCV_STR,50$>,-           ; Start message - process it
                     078C  2142                 <LEV$C_LOG_NFE,50$>,-           ; If error detected, log event
                     078C  2143                 <LEV$C_LOG_ADE,50$>,-           ;       ''
                     078C  2144                 <LEV$C_LOG_CDE,50$>>            ;       ''
        2C   11      07CB  2145 40$:    BRB     IGNORE_MSG                      ; Otherwise, ignore the message
             05      07CD  2146 50$:    RSB                                     ; Return to queue the event
                     07CE  2147
                     07CE  2148         ;
                     07CE  2149         ;   Router Hello message - process new router adjacency
                     07CE  2150         ;
        04   E1      07CE  2151 10$:    BBC     #LPD$V_RUN,-                    ; Skip if circuit not in RUN state
  26 22 A6           07D0  2152                 LPD$W_STS(R6),IGNORE_MSG
      1613   30      07D3  2153         BSBW    BRA_UP                          ; Broadcast router is up
                     07D6  2154                                                 ; Reset R7 to point to new ADJ block
     20 50   E9      07D6  2155         BLBC    R0,IGNORE_MSG                   ; If cannot allocate, then forget it
20 A5    58   B0      07D9  2156         MOVW    R8,WQE$W_ADJ_INX(R5)            ; Store new ADJ index
             05      07DD  2157         RSB                                     ; Queue event set by UNK
                     07DE  2158         ;
                     07DE  2159         ;   Endnode Hello message - process new endnode adjacency
                     07DE  2160         ;
        04   E1      07DE  2161 20$:    BBC     #LPD$V_RUN,-                    ; Skip if circuit not in RUN state
  16 22 A6           07E0  2162                 LPD$W_STS(R6),IGNORE_MSG
      16FA   30      07E3  2163         BSBW    BEA_UP                          ; Broadcast endnode is up
                     07E6  2164                                                 ; Reset R7 to point to new ADJ block
    10 50    E9      07E6  2165         BLBC    R0,IGNORE_MSG                   ; If cannot allocate, then forget it
20 A5    58   B0      07E9  2166         MOVW    R8,WQE$W_ADJ_INX(R5)            ; Store new ADJ index
                     07ED  2167         $LOG    TPL_AUP,,,R5                    ; Set ''adjacency up'' event
      F808'  30      07F5  2168         BSBW    NET$EVT_INTRAW                  ; Log the event record
             05      07F8  2169         RSB                                     ; Queue event set by UNK
                     07F9  2170
                     07F9  2171 ;
                     07F9  2172 ;   This routine is called when we have received a message which
                     07F9  2173 ;   is valid when a adjacency is normally up, but which must be
                     07F9  2174 ;   ignored when the adjacency is still undergoing initialization.
                     07F9  2175 ;
                     07F9  2176
                     07F9  2177 IGNORE_MSG:
          50   D4      07F9  2178         CLRL    R0                              ; Do not queue any event
               05      07FB  2179         RSB
                     07FC  2180
                     07FC  2181 ;
                     07FC  2182 ;   Determine the type of message received, dispatch to parse it
                     07FC  2183 ;
                     07FC  2184
02   61   02   00    ED  07FC  2185 UNK:    CMPZV   #0,#2,(R1),#TR3C_MSG_RTH ; Phase III/IV Data Packet?
             F6   13  0801  2186         BEQL    IGNORE_MSG                      ; If so, drop message on the floor
 59   000000B4'EF    9E  0803  2187         MOVAB   MSG_MAP_TABLE,R9                ; Setup the message mapping table ptr
          53   81    9A  080A  2188         MOVZBL  (R1)+,R3                        ; Get the message type
```

```
        58 8F    53   91  080D          CMPB    R3,#TR2C_MSG_INI          ; Is this a Phase II init message?
                  05   12  0811          BNEQ    80$                       ; If not, branch
                  71   95  0813          TSTB    -(R1)                     ; Backup
           53     81   3C  0815          MOVZWL  (R1)+,R3                  ; Get the type and subtype
           50     89   D0  0818  80$:    MOVL    (R9)+,R0                  ; Get message parser routine address
                  32   13  081B          BEQL    PFE                       ; If at end of table, log error
           52     89   3C  081D          MOVZWL  (R9)+,R2                  ; Get minimum msg size
           89     53   B1  0820          CMPW    R3,(R9)+                  ; Is this it ?
                  F3   12  0823          BNEQ    80$                       ; If not, loop
        16 A5     52   A2  0825          SUBW    R2,WQE$L_PM2+2(R5)        ; Update bytes left
                  24   19  0829          BLSS    PFE                       ; If LSS then packet format error
                  60   16  082B          JSB     (R0)                      ; Parse the message
                       05  082D          RSB
                           082E
                           082E  ;
                           082E  ;   NETDRIVER messages which cause a DECnet event record to be written
                           082E  ;   and the adjacency to be shutdown.
                           082E  ;
                           082E
                           082E  LSN:                                      ; Adjacncy listener timeout
                           082E  ;
                           082E  ;   If this is a DMC line, then we will toggle the line as well as the
                           082E  ;   circuit to force any modem connections, to hang up. This is done
                           082E  ;   here because the DMC driver cannot figure out on it's own when to
                           082E  ;   hang up the modem.
                           082E  ;
        50    28 A6    9A  082E          MOVZBL  LPD$B_PLVEC(R6),R0        ; Get PLVEC index
  01  00000000'EF40    91  0832          CMPB    PLVEC$AB_DEV[R0],-        ; Is this a DMC11?
                           083A                  #DEVTRN$C_DEV_DMC         ;
                  06   12  083A          BNEQ    10$                       ; Br if no
           1000 8F     A8  083C          BISW    #LPD$M_TOGGLE,-           ; Else, force a line toggle
             22 A6         0840                  LPD$W_STS(R6)             ;
             18 A5     D4  0842  10$:    CLRL    WQE$L_EVL_PKT(R5)         ; Indicate no packet for this event
                           0845          $LOG    TPL_LDS,TPL_PRSN_LTMO,,R5 ; Store logging info in WQE
                  13   11  084D          BRB     ADJ_DOWN_EVENT
                           084F  PFE:                                      ; Packet format error
                           084F          BUMP    B,RCB$B_CNT_PFE(R4)       ; Increment packet format error count
                           085A          $LOG    TPL_PFM,,,R5              ; Store logging info in WQE
                           0862  ADJ_DOWN_EVENT:
        10 A5     24   90  0862          MOVB    #LEV$C_LOG_ADE,WQE$B_EVT(R5) ; Setup to log event
           50     01   D0  0866          MOVL    #1,R0                     ; Process event
                       05  0869          RSB                               ; Return true - process event
                           086A
                           086A  ;
                           086A  ;   NETDRIVER messages which simply cause a DECNET event record to be written.
                           086A  ;
                           086A
                           086A  OPL:                                      ; Oversized packet loss
                           086A          $LOG    TPL_OPL,,,R5              ; Store logging info into WQE
                  3D   11  0872          BRB     NON_FATAL                 ; Take common exit
                           0874  APL:                                      ; Aged packet loss
                           0874          BUMP    B,RCB$B_CNT_APL(R4)       ; Increment aged packet loss count
                           087F          $LOG    TPL_APL,,,R5              ; Store logging info in WQE
                  28   11  0887          BRB     NON_FATAL                 ; Take common exit
                           0889  NUL:                                      ; Node unreachable packet loss
                           0889          BUMP    W,RCB$W_CNT_NUL(R4)       ; Increment node unreachable loss count
                           0894          $LOG    TPL_UPL,,,R5              ; Store logging info in WQE
```

```
                     13   11  089C  2246                BRB     NON_FATAL                       ; Take common exit
                                089E  2247 NOL:                                                 ; Node out-of-range packet loss
                                089E  2248                BUMP    B,RCB$B_CNT_NOL(R4)           ; Increment node out of range loss count
                                08A9  2249                $LOG    TPL_RPL,,,R5                  ; Store logging info in WQE
                                08B1  2250 NON_FATAL:                                           ; Common non-fatal event exit
                10 A5    22   90  08B1  2251                MOVB    #LEV$C_LOG_NFE,WQE$B_EVT(R5) ; Setup for "log non-fatal event"
                   50    01   D0  08B5  2252                MOVL    #1,R0                         ; Process event
                                05  08B8  2253                RSB                             ; Return true - process event
                                    08B9  2254
```

```
                        08B9  2256          .SBTTL  RCV_STR2 - Received Phase II start message
                        08B9  2257 ;+
                        08B9  2258 ; RCV_STR2  - Process received Phase II Transport Initialization Start message
                        08B9  2259 ;
                        08B9  2260 ; FUNCTIONAL DESCRIPTION:
                        08B9  2261 ;
                        08B9  2262 ; The message is parsed to determine correctness, node address, and the
                        08B9  2263 ; database is checked to determine whether a verification message needs to be
                        08B9  2264 ; sent.
                        08B9  2265 ;
                        08B9  2266 ; The possible events returned in WQE$B_EVT are:
                        08B9  2267 ;
                        08B9  2268 ;               LEV$C_RCV_STR   - Rcv Transport Layer "start" msg
                        08B9  2269 ;               LEV$C_LOG_FTE   - Fatal event
                        08B9  2270 ;
                        08B9  2271 ;
                        08B9  2272 ; INPUTS:       R11     CNR address
                        08B9  2273 ;               R10     CNF address
                        08B9  2274 ;               R7      ADJ address
                        08B9  2275 ;               R6      LPD address
                        08B9  2276 ;               R5      WQE address
                        08B9  2277 ;               R4      RCB address
                        08B9  2278 ;               R1      Ptr to next byte in the message
                        08B9  2279 ;
                        08B9  2280 ;               All others are scratch
                        08B9  2281 ;
                        08B9  2282 ; OUTPUTS:      R5      Unchanged
                        08B9  2283 ;               R0      True if event to be processed, false if not
                        08B9  2284 ;
                        08B9  2285 ;               All other regs may be clobbered.
                        08B9  2286 ;
                        08B9  2287 ;-
                        08B9  2288 RCV_STR2:                                    ; Process rcvd phase II Start msg
                        08B9  2289 ;
                        08B9  2290 ;   Parse the node address.
                        08B9  2291 ;
     03CF    30         08B9  2292          BSBW    PARSE_PH2_ADDR              ; Parse Phase II node address
   4F 50     E9         08BC  2293          BLBC    R0,50$                      ; If LBC error, chain to event setup
                        08BF  2294                                              ; by PARSE_PH2_ADDR
                        08BF  2295 ;
                        08BF  2296 ;   Process the nodename field.  The size is checked but the name text
                        08BF  2297 ;   itself is ignored (this is consistent with not knowing the name of
                        08BF  2298 ;   a Phase III node and allows the rules for Phase II and Phase III
                        08BF  2299 ;   nodes to be the same with respect to whether or not there needs
                        08BF  2300 ;   to be an NDI in the database for that node -- i.e., an NDI is
                        08BF  2301 ;   needed only if "verification" is required for the circuit which
                        08BF  2302 ;   connects to the node).
                        08BF  2303 ;
     50    81    9A     08BF  2304          MOVZBL  (R1)+,R0                    ; Get bytes in node name
 16  A5    50    A2     08C2  2305          SUBW    R0,WQE$L_PM2+2(R5)          ; Account for them
           4A    19     08C6  2306          BLSS    100$                        ; If LSS then msg is too small
     51    50    C0     08C8  2307          ADDL    R0,R1                       ; Advance past name
                        08CB  2308 ;
                        08CB  2309 ;   Ignore the FUNCTIONS field
                        08CB  2310 ;
           81    95     08CB  2311 20$:     TSTB    (R1)+                       ; If LSS then field is extended
           07    18     08CD  2312          BGEQ    22$                         ; If GEQ then okay
```

```
                16 A5   B7  08CF  2313          DECW    WQE$L_PM2+2(R5)          ; Account for next (optional) byte
                   3E   19  08D2  2314          BLSS    100$                     ; If LSS then format error
                   F5   11  08D4  2315          BRB     20$                      ; Loop
                            08D6  2316 22$:     ;
                            08D6  2317          ;   Process the REQUESTS field
                            08D6  2318          ;
             07 61    00   E1  08D6  2319          BBC     #TR2V_REQ_VRF,(R1),25$   ; If LBC verification not required
                            08DA  2320          SETBIT  LPD$V_XMT_VRF,XMTFLG     ; Indicate verification required
                   81   95  08E1  2321 25$:     TSTB    (R1)+                    ; If LSS then field is extended
                   07   18  08E3  2322          BGEQ    27$                      ; If GEQ then okay
                16 A5   B7  08E5  2323          DECW    WQE$L_PM2+2(R5)          ; Account for next (optional) byte
                   28   19  08E8  2324          BLSS    100$                     ; If LSS then format error
                   F5   11  08EA  2325          BRB     25$                      ; Loop
                            08EC  2326 27$:     ;
                            08EC  2327          ;   Get the partner's block size and version
                            08EC  2328          ;
00000018'EF        81   B0  08EC  2329          MOVW    (R1)+,LEV_W_BLKSIZE      ; Save partner's block size
                   81   D5  08F3  2330          TSTL    (R1)+                    ; Skip over partner's NSP block size
                            08F5  2331                                           ; and his MAX LINKS specifier
00000038'EF        02   90  08F5  2332          MOVB    #ADJ$C_PTY_PH2,PTYPE     ; Mark adjacent node as Phase II
00000000'EF        81   B0  08FC  2333          MOVW    (R1)+,NET$GL_INITVER     ; Save INIT version (3 bytes)
00000002'EF        81   90  0903  2334          MOVB    (R1)+,NET$GL_INITVER+2
              10 A5    08   90  090A  2335          MOVB    #LEV$C_RCV_STR,WQE$B_EVT(R5) ; Event is 'rcvd start msg'
                 50    01   D0  090E  2336 50$:     MOVL    #1,R0                    ; Process event
                            05  0911  2337          RSB                              ; Return true - process event
                            0912  2338
                    FF3A    31  0912  2339 100$:    BRW     PFE                      ; Packet format error
```

```
                                  0915          .SBTTL  RCV_STR3 - Received Phase III start message
                                  0915  ;+
                                  0915  ; RCV_STR3  - Process received Phase II Transport Initialization Start message
                                  0915  ;
                                  0915  ; FUNCTIONAL DESCRIPTION:
                                  0915  ;
                                  0915  ; The message is parsed to determine correctness, node address, and the
                                  0915  ; database is checked to determine whether a verification message needs to be
                                  0915  ; sent.
                                  0915  ;
                                  0915  ; INPUTS:       R11     CNR address
                                  0915  ;               R10     CNF address
                                  0915  ;               R7      ADJ address
                                  0915  ;               R6      LPD address
                                  0915  ;               R5      WQE address
                                  0915  ;               R4      RCB address
                                  0915  ;               R1      Ptr to next byte in the message
                                  0915  ;
                                  0915  ;               All others are scratch
                                  0915  ;
                                  0915  ; OUTPUTS:      R5      Unchanged
                                  0915  ;               R0      True if event to be processed, false if not
                                  0915  ;
                                  0915  ;               All other regs may be clobbered.
                                  0915  ;
                                  0915  ;-
                                  0915  RCV_STR3:                                   ; Process rcvd phase III/IV Start msg
                                  0915          ;
                                  0915          ;   Compare version numbers.  If we receive a start from a node
                                  0915          ;   with a higher version number, then drop the message.  The
                                  0915          ;   other node will detect that we are lower version and re-send
                                  0915          ;   the correct start message.  If the version is lower than ours,
                                  0915          ;   but we don't recognize or support it, then log "version skew".
                                  0915          ;
                     51     DD    0915          PUSHL   R1                          ; Save current pointer
          51     05 A1     9E    0917          MOVAB   5(R1),R1                    ; Point to version field
                  03E6     30    091B          BSBW    PARSE_VERSION               ; Parse version number field
                     51  8ED0    091E          POPL    R1                          ; Restore current pointer
     0301 8F     05 A1     B1    0921          CMPW    5(R1),#TR3C_TIVER           ; Is it Phase III version?
                     05     13    0927          BEQL    5$                          ; If so, override error - we can handle it
                  41 50     E9    0929          BLBC    R0,30$                      ; If error, chain to new event
                     43     11    092C          BRB     RCV_STR4                    ; If no error, then Phase IV
                                  092E          ;
                                  092E          ;   Parse the Phase III start message.
                                  092E          ;
                  0387     30    092E  5$:     BSBW    PARSE_PH3_ADDR              ; Parse phase III node address field
                  39 50     E9    0931          BLBC    R0,30$                      ; Br on error with new event setup by
                                  0934                                              ; PARSE_PH3_ADDR
          07 61     02     E1    0934          BBC     #TR3V_REQ_VRF,(R1),10$      ; Br unless verification is requested
                                  0938          SETBIT  LPD$V_XMT_VRF,XMTFLG        ; Need to send verif. msg
                     00     EF    093F  10$:    EXTZV   #TR3V_REQ_NTY,-             ; Get node type
        50     81     02          0941                  #TR3S_REQ_NTY,(R1)+,R0      ;
   00000038'EF     00     90    0944          MOVB    #ADJ$C_PTY_PH3,PTYPE        ; Assume Phase III routing
              02     50     91    094B          CMPB    R0,#TR3C_NTY_PH3            ; Is it a routing node?
                     0F     13    094E          BEQL    20$                         ; If EQL yes, continue
   00000038'EF     01     90    0950          MOVB    #ADJ$C_PTY_PH3N,PTYPE       ; Assume Phase III non-routing
              03     50     91    0957          CMPB    R0,#TR3C_NTY_PH3N           ; Is it a non-routing node?
```

```
                03   13  095A  2398          BEQL    20$                             ; If EQL yes, continue
              FEF0   31  095C  2399          BRW     PFE                             ; Else report "packet format error"
00000018'EF     81   B0  095F  2400 20$:     MOVW    (R1)+,LEV_W_BLKSIZE             ; Store partner's block size
          51    03   C0  0966  2401          ADDL    #3,R1                           ; Skip version field
       10 A5    08   90  0969  2402          MOVB    #LEV$C_RCV_STR,WQE$B_EVT(R5) ; Event is 'rcvd start msg'
          50    01   D0  096D  2403 30$:     MOVL    #1,R0                           ; Process event
                     05  0970  2404          RSB                                     ; Return true - process event
```

```
                     0971  2406          .SBTTL  RCV_STR4 - Received Phase IV start message
                     0971  2407 ;+
                     0971  2408 ; RCV_STR4  - Process received Phase IV Transport Initialization Start message
                     0971  2409 ;
                     0971  2410 ; FUNCTIONAL DESCRIPTION:
                     0971  2411 ;
                     0971  2412 ; The message is parsed to determine correctness, node address, and the
                     0971  2413 ; database is checked to determine whether a verification message needs to be
                     0971  2414 ; sent.
                     0971  2415 ;
                     0971  2416 ; INPUTS:        R11     CNR address
                     0971  2417 ;                R10     CNF address
                     0971  2418 ;                R7      ADJ address
                     0971  2419 ;                R6      LPD address
                     0971  2420 ;                R5      WQE address
                     0971  2421 ;                R4      RCB address
                     0971  2422 ;                R1      Ptr to next byte in the message
                     0971  2423 ;
                     0971  2424 ;                All others are scratch
                     0971  2425 ;
                     0971  2426 ; OUTPUTS:       R5      Unchanged
                     0971  2427 ;                R0      True if event to be processed, false if not
                     0971  2428 ;
                     0971  2429 ;                All other regs may be clobbered.
                     0971  2430 ;
                     0971  2431 ;-
                     0971  2432 RCV_STR4:                                    ; Process rcvd phase IV Start msg
           0344   30 0971  2433          BSBW    PARSE_PH4_ADDR              ; Parse phase IV node address field
         54 50    E9 0974  2434          BLBC    R0,30$                      ; Br on error with new event setup by
                     0977  2435                                              ; PARSE_PH4_ADDR
       07 61      02 E1 0977  2436          BBC     #TR4V_REQ_VRF,(R1),10$      ; Br unless verification is requested
                     097B  2437          SETBIT  LPD$V_XMT_VRF,XMTFLG        ; Need to send verif. msg
                  00 EF 0982  2438 10$:    EXTZV   #TR3V_REQ_NTY,-             ; Get node type
             50   81 02      0984  2439                  #TR3S_REQ_NTY,(R1)+,R0 ;
00000038'EF      04 90 0987  2440          MOVB    #ADJ$C_PTY_PH4,PTYPE        ; Assume Phase IV routing
            02   50 91 098E  2441          CMPB    R0,#TR4C_NTY_ROU            ; Is it a routing node?
                  1B 13 0991  2442          BEQL    20$                         ; Branch if so
00000038'EF      05 90 0993  2443          MOVB    #ADJ$C_PTY_PH4N,PTYPE       ; Assume Phase IV non-routing
            03   50 91 099A  2444          CMPB    R0,#TR4C_NTY_NROU           ; Is it a non-routing node?
                  0F 13 099D  2445          BEQL    20$                         ; If EQL yes, continue
00000038'EF      03 90 099F  2446          MOVB    #ADJ$C_PTY_AREA,PTYPE       ; Assume Phase IV area routing
            01   50 91 09A6  2447          CMPB    R0,#TR4C_NTY_ARO            ; Is it area-router?
                  03 13 09A9  2448          BEQL    20$                         ; Branch if so
                FEA1 31 09AB  2449          BRW     PFE                         ; Else report "packet format error"
00000018'EF      81 B0 09AE  2450 20$:     MOVW    (R1)+,LEV_W_BLKSIZE         ; Store partner's block size
            51   03 C0 09B5  2451          ADDL    #3,R1                       ; Skip version field
              16 A5 B5 09B8  2452          TSTW    WQE$L_PM2+2(R5)             ; && Was msg exactly 10 bytes?
                  03 12 09BB  2453          BNEQ    25$                         ; && If so,
              01 A1 94 09BD  2454          CLRB    1(R1)                       ; && Clear high order byte for those
                     09C0  2455                                              ; && impl. who onlyused 1 byte hello
00000020'EF      81 B0 09C0  2456 25$:     MOVW    (R1)+,LEV_W_HELLO           ; Store partner's hello timer
            10 A5 08 90 09C7  2457          MOVB    #LEV$C_RCV_STR,WQE$B_EVT(R5) ; Event is 'rcvd start msg'
            50   01 D0 09CB  2458 30$:     MOVL    #1,R0                       ; Process event
                  05 09CE  2459          RSB                                 ; Return true - process event
```

```
                                        09CF  2461          .SBTTL  RCV_VRF - Received routing verification message
                                        09CF  2462 ;+
                                        09CF  2463 ; RCV_VRF2 - Process received Transport Phase II Verification message
                                        09CF  2464 ; RCV_VRF3 - Process received Transport Phase III Verification message
                                        09CF  2465 ; RCV_VRF4 - Process received Transport Phase IV Verification message
                                        09CF  2466 ;
                                        09CF  2467 ; FUNCTIONAL DESCRIPTION:
                                        09CF  2468 ;
                                        09CF  2469 ;
                                        09CF  2470 ; INPUTS:        R11     CNR address
                                        09CF  2471 ;                R10     CNF address
                                        09CF  2472 ;                R7      ADJ address
                                        09CF  2473 ;                R6      LPD address
                                        09CF  2474 ;                R5      WQE address
                                        09CF  2475 ;                R4      RCB pointer
                                        09CF  2476 ;                R1      Ptr to next byte in the message
                                        09CF  2477 ;
                                        09CF  2478 ;                All others are scratch
                                        09CF  2479 ;
                                        09CF  2480 ; OUTPUTS:       R5      Unchanged
                                        09CF  2481 ;                R0      True if event to be processed, false if not
                                        09CF  2482 ;
                                        09CF  2483 ;                All other registers may be clobbered.
                                        09CF  2484 ;
                                        09CF  2485 ;-
                                        09CF  2486          .ENABL  LSB
                                        09CF  2487
                                        09CF  2488 RCV_VRF2:                                    ; Preprocess rcv'd Phase II Verf msg
00000038'EF    02    90                 09CF  2489          MOVB    #ADJ$C_PTY_PH2,PTYPE        ; Mark node is Phase II
              08    B1                 09D6  2490          CMPW    S^#TR2C_PSW_LNG,-           ; Is the msg size correct?
          16 A5                        09D8  2491                  WQE$L_PM2+2(R5)             ;
              18    13                 09DA  2492          BEQL    5$                          ; If EQL yes, save password
            FE70    31                 09DC  2493          BRW     PFE                         ; Else report "packet format error"
                                        09DF  2494
                                        09DF  2495 RCV_VRF3:
                                        09DF  2496 RCV_VRF4:
              02D6    30                09DF  2497          BSBW    PARSE_PH3_ADDR              ; Get partner's address
            22 50     E9                09E2  2498          BLBC    R0,10$                      ; If LBC error, exit with event setup
                                        09E5  2499                                              ; by PARSE_PH3_ADDR
       50     81     9A                09E5  2500          MOVZBL  (R1)+,R0                    ; Get count of password text
    16 A5     50     B1                09E8  2501          CMPW    R0,WQE$L_PM2+2(R5)          ; Does it match bytes left?
              1D     12                09EC  2502          BNEQ    20$                         ; If not, illegal message
    40 8F     50     91                09EE  2503          CMPB    R0,#TR3C_MAX_PSW            ; Is it too large
              17     1A                09F2  2504          BGTRU   20$                         ; If so, illegal message
                                        09F4  2505          ;
                                        09F4  2506          ;   Store the password descriptor
                                        09F4  2507          ;
 10 A5     09     90                    09F4  2508 5$:      MOVB    #LEV$C_RCV_VRF,WQE$B_EVT(R5)    ; Setup event code
00000024'EF  16 A5     9A             09F8  2509          MOVZBL  WQE$L_PM2+2(R5),LEV_Q_PSWDESC   ; Save password size
   00000028'EF     51    D0           0A00  2510          MOVL    R1,LEV_Q_PSWDESC+4              ; Save password pointer
          50     01    D0             0A07  2511 10$:     MOVL    #1,R0                       ; Process event
                       05             0A0A  2512          RSB                                 ; Return true - process event
                                      0A0B  2513
            FE41    31                0A0B  2514 20$:     BRW     PFE                         ; Report "packet format error"
                                      0A0E  2515
                                      0A0E  2516          .DSABL  LSB
```

```
                     0A0E  2518          .SBTTL  RCV_RHEL - Received Phase IV Router Hello message
                     0A0E  2519 ;+
                     0A0E  2520 ; RCV_RHEL  - Process received Phase IV NI Router Hello message
                     0A0E  2521 ;
                     0A0E  2522 ; FUNCTIONAL DESCRIPTION:
                     0A0E  2523 ;
                     0A0E  2524 ; The message is parsed and validated, and an event is queued indicating
                     0A0E  2525 ; that the message needs processing.
                     0A0E  2526 ;
                     0A0E  2527 ; INPUTS:        R11     CNR address
                     0A0E  2528 ;                R10     CNF address
                     0A0E  2529 ;                R7      ADJ address
                     0A0E  2530 ;                R6      LPD address
                     0A0E  2531 ;                R5      WQE address
                     0A0E  2532 ;                R4      RCB address
                     0A0E  2533 ;                R1      Ptr to next byte in the message
                     0A0E  2534 ;
                     0A0E  2535 ;                All others are scratch
                     0A0E  2536 ;
                     0A0E  2537 ; OUTPUTS:       R5      Unchanged
                     0A0E  2538 ;                R0      True if event to be processed, false if not
                     0A0E  2539 ;
                     0A0E  2540 ;                All other regs may be clobbered.
                     0A0E  2541 ;-
                     0A0E  2542 RCV_RHEL:                                ; Process rcvd Phase IV Router Hello
                     0A0E  2543                 ;
                     0A0E  2544                 ;   Compare version numbers.  If we receive a message from a node
                     0A0E  2545                 ;   with a higher version number, then drop the message.  The
                     0A0E  2546                 ;   other node will detect that we are lower version and re-send
                     0A0E  2547                 ;   the correct message.  If the version is lower than ours,
                     0A0E  2548                 ;   but we don't recognize or support it, then log "version skew".
                     0A0E  2549                 ;
           02F3   30 0A0E  2550                 BSBW    PARSE_VERSION       ; Parse the version field
         76 50    E9 0A11  2551                 BLBC    R0,30$              ; If error, chain to new event
                     0A14  2552                 ;
                     0A14  2553                 ;   Parse the remote node's address
                     0A14  2554                 ;
000400AA 8F    81    D1 0A14  2555                 CMPL    (R1)+,#TR$C_NI_PREFIX ; Standard NI prefix?
               71    12 0A1B  2556                 BNEQ    70$                 ; Ignore msg if not
             0298    30 0A1D  2557                 BSBW    PARSE_PH4_ADDR      ; Parse phase IV node address field
         67 50    E9 0A20  2558                 BLBC    R0,30$              ; Br on error with new event setup by
                     0A23  2559                 ;                           ; PARSE_PH4_ADDR
                     0A23  2560                 ;
                     0A23  2561                 ;   Parse the node type code
                     0A23  2562                 ;
            00    EF 0A23  2563                 EXTZV   #TR3V_REQ_NTY,-     ; Get node type
     50     02       0A25  2564                         #TR3S_REQ_NTY,(R1)+,R0 ;
00000038'EF 04    90 0A28  2565                 MOVB    #ADJ$C_PTY_PH4,PTYPE ; Assume Phase IV routing
            02    50 91 0A2F  2566                 CMPB    R0,#TR4C_NTY_ROU    ; Is it a routing node?
               0F    13 0A32  2567                 BEQL    20$                 ; Branch if so
00000038'EF 03    90 0A34  2568                 MOVB    #ADJ$C_PTY_AREA,PTYPE ; Assume Phase IV area routing
            01    50 91 0A3B  2569                 CMPB    R0,#TR4C_NTY_ARO    ; Is it area-router?
               03    13 0A3E  2570                 BEQL    20$                 ; Branch if so
             FE0C    31 0A40  2571                 BRW     PFE                 ; Else report "packet format error"
                     0A43  2572 20$:               ;
                     0A43  2573                    ;   If this is a message from a node in another area, then ignore
                     0A43  2574                    ;   it unless then node is also a level 2 router.  We assume that
```

```
                                0A43          ;   the only way to get a message from another area past the address
                                0A43          ;   parsing routine is for us to be a level 2 router.
                                0A43          ;
                                0A43          ;   This essentially allows level2-level2 connections, but
                                0A43          ;   disallows level2-level1 connections over the NI.
                                0A43          ;
                    0A    EF    0A43          EXTZV   #TR4$V_ADDR_AREA,-          ; Get area number of sending node
          50    58    06        0A45                  #TR4$S_ADDR_AREA,R8,R0
          008B  C4    50    91  0A48          CMPB    R0,RCB$B_HOMEAREA(R4)       ; Our area?
                    09    13  0A4D          BEQL    22$                         ; If not, drop the message
    03    00000038'EF     91  0A4F          CMPB    PTYPE,#ADJ$C_PTY_AREA       ; Is the remote node a level 2 router?
                    36    12  0A56          BNEQ    70$                         ; If not, ignore the message
                                0A58  22$:    ;
                                0A58          ;   Parse remaining fields
                                0A58          ;
    00000018'EF     81    B0  0A58          MOVW    (R1)+,LEV_W_BLKSIZE         ; Store partner's block size
    0000001C'EF     81    90  0A5F          MOVB    (R1)+,LEV_B_PRIORITY        ; Store router priority
                    51    D6  0A66          INCL    R1                          ; Skip AREA reserved field
    00000020'EF     81    B0  0A68          MOVW    (R1)+,LEV_W_HELLO           ; Store partner's hello timer
                    07    12  0A6F          BNEQ    25$                         ; && If not filled in, assume old impl.
    00000020'EF     61    9B  0A71          MOVZBW  (R1),LEV_W_HELLO            ; && who still used 1 byte hello
              51    09    C0  0A78  25$:    ADDL    #1+1+7,R1                   ; Skip reserved, count byte, LOGICAL NAME
          16  A5    81    9B  0A7B          MOVZBW  (R1)+,WQE$L_PM2+2(R5)       ; Store size of R/S LIST
              51    55    C2  0A7F          SUBL    R5,R1                       ; Compute offset to R/S LIST
          14  A5    51    B0  0A82          MOVW    R1,WQE$L_PM2(R5)            ; Store offset to list
          10  A5    0D    90  0A86          MOVB    #LEV$C_RCV_RHEL,WQE$B_EVT(R5) ; Event is 'rcvd Router Hello'
              50    01    D0  0A8A  30$:    MOVL    #1,R0                       ; Process event
                    05        0A8D          RSB                                 ; Return true - process event
                                0A8E
                                0A8E  ;
                                0A8E  ; Drop the message on the floor.
                                0A8E  ;
                                0A8E
                    50    D4  0A8E  70$:    CLRL    R0                          ; Return false - do not queue event
                    05        0A90          RSB
```

```
                     0A91  2611          .SBTTL  RCV_EHEL - Received Phase IV Endnode Hello message
                     0A91  2612 ;+
                     0A91  2613 ; RCV_EHEL  - Process received Phase IV NI Endnode Hello message
                     0A91  2614 ;
                     0A91  2615 ; FUNCTIONAL DESCRIPTION:
                     0A91  2616 ;
                     0A91  2617 ; The message is parsed and validated, and an event is queued indicating
                     0A91  2618 ; that the message needs processing.
                     0A91  2619 ;
                     0A91  2620 ; INPUTS:        R11     CNR address
                     0A91  2621 ;                R10     CNF address
                     0A91  2622 ;                R7      ADJ address
                     0A91  2623 ;                R6      LPD address
                     0A91  2624 ;                R5      WQE address
                     0A91  2625 ;                R4      RCB address
                     0A91  2626 ;                R1      Ptr to next byte in the message
                     0A91  2627 ;
                     0A91  2628 ;                All others are scratch
                     0A91  2629 ;
                     0A91  2630 ; OUTPUTS:       R5      Unchanged
                     0A91  2631 ;                R0      True if event to be processed, false if not
                     0A91  2632 ;
                     0A91  2633 ;                All other regs may be clobbered.
                     0A91  2634 ;-
                     0A91  2635 RCV_EHEL:                                ; Process rcvd Phase IV Endnode Hello
                     0A91  2636                 ;
                     0A91  2637                 ;   Compare version numbers.  If we receive a message from a node
                     0A91  2638                 ;   with a higher version number, then drop the message.  The
                     0A91  2639                 ;   other node will detect that we are lower version and re-send
                     0A91  2640                 ;   the correct message.  If the version is lower than ours,
                     0A91  2641                 ;   but we don't recognize or support it, then log "version skew".
                     0A91  2642                 ;
           0270   30  0A91  2643          BSBW    PARSE_VERSION          ; Parse the version field
         4A 50    E9  0A94  2644          BLBC    R0,30$                 ; If error, chain to new event
                      0A97  2645                 ;
                      0A97  2646                 ;   Parse the Endnode Hello message
                      0A97  2647                 ;
000400AA 8F  81    D1  0A97  2648          CMPL    (R1)+,#TR$C_NI_PREFIX  ; Standard NI prefix?
               45    12  0A9E  2649          BNEQ    70$                    ; Ignore msg if not
             0215    30  0AA0  2650          BSBW    PARSE_PH4_ADDR         ; Parse phase IV node address field
          3B 50    E9  0AA3  2651          BLBC    R0,30$                 ; Br on error with new event setup by
                      0AA6  2652                                         ; PARSE_PH4_ADDR
               0A    EF  0AA6  2653          EXTZV   #TR4$V_ADDR_AREA,-     ; Get area number of sending node
   50    58    06        0AA8  2654                  #TR4$S_ADDR_AREA,R8,R0
  008B C4      50    91  0AAB  2655          CMPB    R0,RCB$B_HOMEAREA(R4)  ; Our area?
               33    12  0AB0  2656          BNEQ    70$                    ; If not, drop the message
               00    ED  0AB2  2657          CMPZV   #TR3V_REQ_NTY,-        ; Get node type
         81    02        0AB4  2658                  #TR3S_REQ_NTY,(R1)+,-  ;
               03        0AB6  2659                  #TR4C_NTY_NROU         ; Is it a endnode?
               03    13  0AB7  2660          BEQL    20$                    ; Branch if so
             FD93    31  0AB9  2661          BRW     PFE                    ; Else report "packet format error"
00000038'EF    05    90  0ABC  2662 20$:     MOVB    #ADJ$C_PTY_PH4N,PTYPE  ; Mark Phase IV endnode message
00000018'EF    81    B0  0AC3  2663          MOVW    (R1)+,LEV_W_BLKSIZE    ; Store partner's block size
          51   0F    C0  0ACA  2664          ADDL    #1+8+6,R1              ; Skip AREA, SEED reserved fields
                      0ACD  2665                                         ; Skip NEIGHBOR (designated router)
00000020'EF    81    B0  0ACD  2666          MOVW    (R1)+,LEV_W_HELLO      ; Store partner's hello timer
               07    12  0AD4  2667          BNEQ    25$                    ; && If not filled in, assume old impl.
```

```
00000020'EF   61   9B   0AD6  2668          MOVZBW  (R1),LEV_W_HELLO          ; && who still used 1 byte hello
        10 A5   0E   90   0ADD  2669 25$:     MOVB    #LEV$C_RCV_EHEL,WQE$B_EVT(R5) ; Event is 'rcvd Endnode Hello'
           50   01   D0   0AE1  2670 30$:     MOVL    #1,R0                     ; Process event
                05   0AE4  2671              RSB                               ; Return true - process event
                     0AE5  2672
                     0AE5  2673 ;
                     0AE5  2674 ; Drop the message on the floor.
                     0AE5  2675 ;
                     0AE5  2676
                50   D4   0AE5  2677 70$:     CLRL    R0                        ; Return false - do not queue event
                05   0AE7  2678              RSB
```

```
                                0AE8  2680          .SBTTL  RCV_RT3 - Received Phase III routing message
                                0AE8  2681 ;+
                                0AE8  2682 ; RCV_RT - Routing message received
                                0AE8  2683 ;
                                0AE8  2684 ; FUNCTIONAL DESCRIPTION:
                                0AE8  2685 ;
                                0AE8  2686 ;       Verify the routing message header and checksum and queue a
                                0AE8  2687 ;       routing update event.
                                0AE8  2688 ;
                                0AE8  2689 ; INPUTS:        R11     CNR address
                                0AE8  2690 ;                R10     CNF address
                                0AE8  2691 ;                R7      ADJ address
                                0AE8  2692 ;                R6      LPD address
                                0AE8  2693 ;                R5      WQE address
                                0AE8  2694 ;                R4      RCB pointer
                                0AE8  2695 ;                R1      Ptr to next byte in the message
                                0AE8  2696 ;
                                0AE8  2697 ;                All others are scratch
                                0AE8  2698 ;
                                0AE8  2699 ; OUTPUTS:       R5      Unchanged
                                0AE8  2700 ;                R0      True if event to be processed, false if not
                                0AE8  2701 ;
                                0AE8  2702 ;                All other registers may be clobbered.
                                0AE8  2703 ;
                                0AE8  2704 ;-
                                0AE8  2705 RCV_RT:                                      ; Process a routing message
          00   01 A7   91       0AE8  2706          CMPB    ADJ$B_PTYPE(R7),#ADJ$C_PTY_PH3 ; Phase III or Phase IV?
                 07   13       0AEC  2707          BEQL    RCV_RT3                     ; Branch if Phase III
                 01   A2       0AEE  2708          SUBW    #TR4C_RT_LNG-TR3C_RT_LNG,- ; Adjust length of msg left
               16 A5            0AF0  2709                  WQE$L_PM2+2(R5)           ; for Phase IV message
                0071   31       0AF2  2710          BRW     RCV_RT4                     ; Process Phase IV routing message
                                0AF5  2711
                                0AF5  2712 RCV_RT3:
00000038'EF      00   90       0AF5  2713          MOVB    #ADJ$C_PTY_PH3,PTYPE        ; Indicate type of message
                01B9   30       0AFC  2714          BSBW    PARSE_PH3_ADDR              ; Parse the node address
               40 50   E9       0AFF  2715          BLBC    R0,15$                      ; If LBC then error
     14 A5     51   55   A3       0B02  2716          SUBW3   R5,R1,WQE$L_PM2(R5)       ; Save offset to current msg byte
           59    16 A5   3C       0B07  2717          MOVZWL  WQE$L_PM2+2(R5),R9        ; Get msg bytes remaining
              29 59   E8       0B0B  2718          BLBS    R9,13$                      ; Must be an even number
           59    02   C6       0B0E  2719          DIVL    #2,R9                       ; Get number of words
                 24   13       0B11  2720          BEQL    13$                         ; Illegal msg if EQL
                                0B13  2721          ;
                                0B13  2722          ;   Calculate checksum  --  R9 does not include the checksum.
                                0B13  2723          ;   The highest node address associated with a non-infinite cost/hops
                                0B13  2724          ;   message cell is determined.  If that address is greater than
                                0B13  2725          ;   our current "max address" then it is reported as an event.
                                0B13  2726          ;
              50   01   D0       0B13  2727          MOVL    #1,R0                       ; Setup loop counter
                   52   7C       0B16  2728          CLRQ    R2                          ; Init check sum (R2) and highest node
                                0B18  2729                                              ; (R3) reachable by partner
         7FFF 8F   61   B1       0B18  2730 10$:     CMPW    (R1),#^X<7FFF>              ; Compare to infinite "cost,hops"
                   18   1A       0B1D  2731          BGTRU   13$                         ; If GTRU then field is invalid
                   03   13       0B1F  2732          BEQL    11$                         ; If EQL then not reachable by partner
              53   50   D0       0B21  2733          MOVL    R0,R3                       ; Save highest reachable address
              52   81   A0       0B24  2734 11$:     ADDW    (R1)+,R2                    ; Calculate checksum via 1's complement
              52   00   D8       0B27  2735          ADWC    #0,R2                       ;   add - needs "end around carry"
            EA 50   59   F3       0B2A  2736          AOBLEQ  R9,R0,10$                   ; Loop until all segments processed
```

```
                10 A5    0B   90   0B2E  2737          MOVB    #LEV$C_RCV_RT,WQE$B_EVT(R5) ; Set up event assuming valid checksum
                   61    52   B1   0B32  2738          CMPW    R2,(R1)                     ; Check sum valid ?
                         0B   13   0B35  2739          BEQL    15$                         ; If EQL then  valid
                                   0B37  2740 13$:      $LOG    TPL_LDS,TPL_PRSN_RUCS,,R5 ; due to "routing update checksum"
                       FD20   31   0B3F  2741          BRW     ADJ_DOWN_EVENT              ; Report fatal event
                                   0B42  2742          ;
                                   0B42  2743          ;   The message is okay.  Log "partial routing update loss" if needed.
                                   0B42  2744          ;
                5A A4    53   B1   0B42  2745 15$:     CMPW    R3,RCB$W_MAX_ADDR(R4)       ; Is partner's highest reachable node
                                   0B46  2746                                              ; address within range?
                         1A   1B   0B46  2747          BLEQU   50$                         ; If LEQU then yes
                                   0B48  2748          BUMP    B,RCB$B_CNT_RUL(R4)         ; Inc count for this event
                                   0B53  2749          $LOG    TPL_PRU,,,R5                ; Setup event logging code
                1E A5    53   B0   0B5B  2750          MOVW    R3,WQE$B_EVL_DT1(R5)        ; Store partner's highest reachable node
                       F49E'  30   0B5F  2751          BSBW    NET$EVT_INTRAW              ; Log the event
                   50    01   D0   0B62  2752 50$:     MOVL    #1,R0                       ; Process event
                                   0B65  2753          RSB                                 ; Return true - process event
```

```
                                0B66  2755          .SBTTL  RCV_RT4 - Received Phase IV routing message
                                0B66  2756 ;+
                                0B66  2757 ; RCV_RT4 - Phase IV routing message received
                                0B66  2758 ;
                                0B66  2759 ;       Verify the routing message header and checksum, and queue a
                                0B66  2760 ;       routing update event.
                                0B66  2761 ;
                                0B66  2762 ; Inputs:
                                0B66  2763 ;
                                0B66  2764 ;       R11 = CNR address
                                0B66  2765 ;       R10 = CNF address
                                0B66  2766 ;       R7 = ADJ address
                                0B66  2767 ;       R6 = LPD address
                                0B66  2768 ;       R5 = WQE address
                                0B66  2769 ;       R4 = RCB address
                                0B66  2770 ;       R1 = Pointer to next byte in message
                                0B66  2771 ;
                                0B66  2772 ; Outputs:
                                0B66  2773 ;
                                0B66  2774 ;       R0 = True if event to be queued, false if not
                                0B66  2775 ;-
                                0B66  2776 RCV_RT4:
00000038'EF    04    90    0B66  2777          MOVB    #ADJ$C_PTY_PH4,PTYPE    ; Indicate type of message
              0148    30    0B6D  2778          BSBW    PARSE_PH4_ADDR          ; Parse Phase IV node address
           3E 50    E9    0B70  2779          BLBC    R0,90$                  ; If error, do event setup by parse
             0A     EF    0B73  2780          EXTZV   #TR4$V_ADDR_AREA,-      ; Get area number of sending node
     50    58     06          0B75  2781                  #TR4$S_ADDR_AREA,R8,R0
     008B C4      50    91    0B78  2782          CMPB    R0,RCB$B_HOMEAREA(R4)   ; Our area?
                  36    12    0B7D  2783          BNEQ    70$                     ; If not, drop the message
                  51    D6    0B7F  2784          INCL    R1                      ; Skip reserved byte
 14 A5    51      55    A3    0B81  2785          SUBW3   R5,R1,WQE$L_PM2(R5)     ; Save offset to first segment
       53    16 A5    3C    0B86  2786          MOVZWL  WQE$L_PM2+2(R5),R3      ; Get msg bytes remaining
          2B 53       E8    0B8A  2787          BLBS    R3,80$                  ; If odd, packet format error
             53       02    C6    0B8D  2788          DIVL    #2,R3           ; Get number of words
                      26    13    0B90  2789          BEQL    80$             ; Illegal msg if EQL
                                0B92  2790          ;
                                0B92  2791          ;   Calculate checksum and check it
                                0B92  2792          ;
             52       01    D0    0B92  2793          MOVL    #1,R2           ; Init check sum
             52       81    A0    0B95  2794 10$:     ADDW    (R1)+,R2        ; Calculate checksum via 1's complement
             52       00    D8    0B98  2795          ADWC    #0,R2           ;   add - needs "end around carry"
               F7     53    F5    0B9B  2796          SOBGTR  R3,10$          ; Loop thru all segments
             61       52    B1    0B9E  2797          CMPW    R2,(R1)         ; Check sum valid ?
                      15    12    0BA1  2798          BNEQ    80$             ; If NEQ, then checksum error
                                0BA3  2799          ;
                                0BA3  2800          ;   Check if any routing update loss, and if so, log an event.
                                0BA3  2801          ;
          58    5A A4    3C    0BA3  2802          MOVZWL  RCB$W_MAX_ADDR(R4),R8   ; Set upper limit for rtginfo
                 0068    30    0BA7  2803          BSBW    CHK_ROS4                ; Check for routing update loss
                   0B 50 E9    0BAA  2804          BLBC    R0,80$                  ; Branch if packet format error
                                0BAD  2805          ;
                                0BAD  2806          ;   Accept the message as valid.  Set up event to process it.
                                0BAD  2807          ;
          10 A5    0B    90    0BAD  2808          MOVB    #LEV$C_RCV_RT,WQE$B_EVT(R5) ; Set up event
             50    01    D0    0BB1  2809 90$:     MOVL    #1,R0           ; Process event
                         05    0BB4  2810          RSB                     ; Return true - process event
                                0BB5  2811
```

```
                50   D4  0BB5  2812 70$:    CLRL    R0                          ; Ignore message
                     05  0BB7  2813         RSB
                         0BB8  2814
                         0BB8  2815 ;
                         0BB8  2816 ; Log a "routing checksum" event
                         0BB8  2817 ;
                         0BB8  2818 80$:    $LOG    TPL_LDS,TPL_PRSN_RUCS,,R5 ; due to "routing update checksum"
              FC9F   31  0BC0  2819         BRW     ADJ_DOWN_EVENT              ; Report fatal event
```

```
                     0BC3  2821          .SBTTL  RCV_ART - Area Routing message received
                     0BC3  2822 ;+
                     0BC3  2823 ; RCV_ART - Area Routing message received
                     0BC3  2824 ;
                     0BC3  2825 ; FUNCTIONAL DESCRIPTION:
                     0BC3  2826 ;
                     0BC3  2827 ;       Verify the area routing message header and checksum and queue a
                     0BC3  2828 ;       routing update event.
                     0BC3  2829 ;
                     0BC3  2830 ; INPUTS:       R11     CNR address
                     0BC3  2831 ;               R10     CNF address
                     0BC3  2832 ;               R7      ADJ address
                     0BC3  2833 ;               R6      LPD address
                     0BC3  2834 ;               R5      WQE address
                     0BC3  2835 ;               R4      RCB pointer
                     0BC3  2836 ;               R1      Ptr to next byte in the message
                     0BC3  2837 ;
                     0BC3  2838 ;               All others are scratch
                     0BC3  2839 ;
                     0BC3  2840 ; OUTPUTS:      R5      Unchanged
                     0BC3  2841 ;               R0      True if event to be processed, false if not
                     0BC3  2842 ;
                     0BC3  2843 ;               All other registers may be clobbered.
                     0BC3  2844 ;
                     0BC3  2845 ;-
                     0BC3  2846 RCV_ART:                                     ; Process an area routing message
00000038'EF    03    90  0BC3  2847          MOVB    #ADJ$C_PTY_AREA,PTYPE   ; Indicate type of message
              00EB   30  0BCA  2848          BSBW    PARSE_PH4_ADDR          ; Parse Phase IV node address
           33 50     E9  0BCD  2849          BLBC    R0,90$                  ; Branch if error detected
              51     D6  0BD0  2850          INCL    R1                      ; Skip reserved byte
14 A5      51     55    A3  0BD2  2851          SUBW3   R5,R1,WQE$L_PM2(R5)     ; Save offset to first segment
        53   16 A5    3C  0BD7  2852          MOVZWL  WQE$L_PM2+2(R5),R3      ; Get msg bytes remaining
           29 53     E8  0BDB  2853          BLBS    R3,80$                  ; If odd, packet format error
           53  02     C6  0BDE  2854          DIVL    #2,R3                   ; Get number of words
               24    13  0BE1  2855          BEQL    80$                     ; Illegal msg if EQL
                         0BE3  2856          ;
                         0BE3  2857          ;   Calculate checksum and check it
                         0BE3  2858          ;
               52    01  D0  0BE3  2859          MOVL    #1,R2                   ; Init check sum
               52    81  A0  0BE6  2860 10$:     ADDW    (R1)+,R2                ; Calculate checksum via 1's complement
               52    00  D8  0BE9  2861          ADWC    #0,R2                   ;   add - needs "end around carry"
                 F7  53  F5  0BEC  2862          SOBGTR  R3,10$                  ; Loop thru all segments
               61    52  B1  0BEF  2863          CMPW    R2,(R1)                 ; Check sum valid ?
                     13  12  0BF2  2864          BNEQ    80$                     ; If NEQ, then checksum error
                         0BF4  2865          ;
                         0BF4  2866          ;   Check if any routing update loss, and if so, log an event.
                         0BF4  2867          ;
58     008C  C4    9A    0BF4  2868          MOVZBL  RCB$B_MAX_AREA(R4),R8   ; Set upper limit for rtginfo
              0016   30  0BF9  2869          BSBW    CHK_ROS4                ; Check for routing update loss
           08 50     E9  0BFC  2870          BLBC    R0,80$                  ; Branch if packet format error
                         0BFF  2871          ;
                         0BFF  2872          ;   Accept the message as valid.  Set up event to process it.
                         0BFF  2873          ;
10 A5      0C    90    0BFF  2874          MOVB    #LEV$C_RCV_ART,WQE$B_EVT(R5) ; Set up event
           50    01    D0  0C03  2875 90$:     MOVL    #1,R0                   ; Process event
                       05  0C06  2876          RSB                             ; Return true - process event
                         0C07  2877
```

```
                     0C07  2878 ;
                     0C07  2879 ; Log a "routing checksum" event
                     0C07  2880 ;
                     0C07  2881 80$:    $LOG    TPL_LDS,TPL_PRSN_RUCS,,R5 ; due to "routing update checksum"
FC50   31            0C0F  2882         BRW     ADJ_DOWN_EVENT            ; Report fatal event
```

```
                                M 7
                     0C12  2884          .SBTTL  Check for routing update loss
                     0C12  2885 ;+
                     0C12  2886 ; CHK_RUS4 - Check for routing update loss in Phase IV format messages
                     0C12  2887 ;
                     0C12  2888 ; Inputs:
                     0C12  2889 ;
                     0C12  2890 ;       R5 = WQE address
                     0C12  2891 ;       R4 = RCB address
                     0C12  2892 ;       R8 = Maximum allowed node/area number in message
                     0C12  2893 ;
                     0C12  2894 ; Outputs:
                     0C12  2895 ;
                     0C12  2896 ;       R0 = True if packet scanned successfully, False if format error
                     0C12  2897 ;       Event is logged, if necessary
                     0C12  2898 ;
                     0C12  2899 ;       R1-R3 are destroyed.
                     0C12  2900 ;-
                     0C12  2901 CHK_RUS4:
          03C0 8F    BB   0C12  2902          PUSHR   #^M<R6,R7,R8,R9>        ; Save registers
                     0C16  2903          ;
                     0C16  2904          ;   Compute the highest reachable node in this routing message
                     0C16  2905          ;   to be used to check for routing update loss.  Note that
                     0C16  2906          ;   only segments which contain info for nodes higher than
                     0C16  2907          ;   max address are even checked for the highest reachable
                     0C16  2908          ;   node, as an optimization.
                     0C16  2909          ;
               53    D4   0C16  2910          CLRL    R3                      ; Preset "highest reachable node"
     59   14 A5    3C   0C18  2911          MOVZWL  WQE$L_PM2(R5),R9        ; Get msg offset to routing info
          59 55    C0   0C1C  2912          ADDL    R5,R9                   ; Convert to pointer
     57   16 A5    3C   0C1F  2913          MOVZWL  WQE$L_PM2+2(R5),R7      ; Get number of bytes of rtginfo
          57 04    C2   0C23  2914 50$:     SUBL    #4,R7                   ; Account for COUNT & STARTID
             5F    15   0C26  2915          BLEQ    80$                     ; Branch if packet format error
             51 89 3C   0C28  2916          MOVZWL  (R9)+,R1                ; Get number of nodes in segment
             52 89 3C   0C2B  2917          MOVZWL  (R9)+,R2                ; Get starting node number
     56      51 01 78   0C2E  2918          ASHL    #1,R1,R6                ; Compute number of bytes of rtginfo
             57 56 C2   0C32  2919          SUBL    R6,R7                   ; Account for cost/hops info
                50 19   0C35  2920          BLSS    80$                     ; Branch if packet format error
     50      FF A142 9E   0C37  2921          MOVAB   -1(R1)[R2],R0           ; Compute highest node in segment
             58 50 D1   0C3C  2922          CMPL    R0,R8                   ; Within max address?
                05 1A   0C3F  2923          BGTRU   53$                     ; If within range, skip scanning segment
             59 56 C0   0C41  2924          ADDL    R6,R9                   ; Skip past entire segment
                16 11   0C44  2925          BRB     58$                     ; Continue with next segment
     7FFF    8F 89 B1   0C46  2926 53$:     CMPW    (R9)+,#^X<7FFF>         ; Compare with infinite cost/hops
                3A 1A   0C4B  2927          BGTRU   80$                     ; If GTR, then field is invalid
                08 13   0C4D  2928          BEQL    55$                     ; Branch if not reachable by partner
             53 52 D1   0C4F  2929          CMPL    R2,R3                   ; Is this the highest reachable node?
                03 1B   0C52  2930          BLEQU   55$                     ; Branch if not
             53 52 D0   0C54  2931          MOVL    R2,R3                   ; Else, save highest reachable node
                52 D6   0C57  2932 55$:     INCL    R2                      ; Skip to next node
                EA 51 F5   0C59  2933          SOBGTR  R1,53$                  ; Loop thru all nodes in segment
                57 D5   0C5C  2934 58$:     TSTL    R7                      ; Any more segments?
                C3 14   0C5E  2935          BGTR    50$                     ; If so, continue
                        0C60  2936          ;
                        0C60  2937          ;   If the highest reachable node is greater than our maximum
                        0C60  2938          ;   address, then log a non-fatal event.
                        0C60  2939          ;
             58 53 D1   0C60  2940          CMPL    R3,R8                   ; Greater than max address?
```

```
          1A   1B  0C63  2941          BLEQU   60$                      ; Branch if ok
                   0C65  2942          BUMP    B,RCB$B_CNT_RUL(R4)      ; Inc count for this event
                   0C70  2943          $LOG    TPL_PRU,,,R5             ; Setup event logging code
   1E A5   53   B0  0C78  2944          MOVW    R3,WQE$B_EVL_DT1(R5)     ; Store partner's highest reachable node
        F381'   30  0C7C  2945          BSBW    NET$EVT_INTRAW           ; Log the event
      50   01   D0  0C7F  2946 60$:     MOVL    #1,R0                    ; Packet format is ok
      03C0 8F   BA  0C82  2947 90$:     POPR    #^M<R6,R7,R8,R9>         ; Restore registers
                05  0C86  2948          RSB
                   0C87  2949
          50   D4  0C87  2950 80$:     CLRL    R0                       ; Indicate bad packet format
          F7   11  0C89  2951          BRB     90$                      ; exit
```

```
                                0C8B          .SBTTL  Parse phase II/III/IV address
                                0C8B  ;+
                                0C8B  ; PARSE_PH2_ADDR        - Parse Phase II  address field
                                0C8B  ; PARSE_PH3_ADDR        - Parse Phase III address field
                                0C8B  ; PARSE_PH4_ADDR        - Parse Phase IV  address field
                                0C8B  ;
                                0C8B  ;
                                0C8B  ; INPUTS:       R6      LPD address
                                0C8B  ;               R5      WQE address
                                0C8B  ;               R4      RCB address
                                0C8B  ;               R1      Pointer to next field in message (node address)
                                0C8B  ;               R0      Scratch
                                0C8B  ;
                                0C8B  ; OUTPUTS:      R8      Node address from message
                                0C8B  ;               R1      Advance passed node address field in message
                                0C8B  ;               R0      LBS if successful
                                0C8B  ;                       LBC otherwise.  In this case an event code is setup
                                0C8B  ;                       for the state table processing and the Event Logging
                                0C8B  ;                       info are setup in the WQE.
                                0C8B  ;
                                0C8B  ;               All other regs are unchanged
                                0C8B  ;
                                0C8B  ;-
                                0C8B          .ENABL  LSB
                                0C8B
                                0C8B  PARSE_PH2_ADDR:                                 ; Parse Phase II node address field
                 58   81   98   0C8B          CVTBL   (R1)+,R8                      ; Get partner's node address
                      15   18   0C8E          BGEQ    10$                           ; Br unless field is extended
                 50   81   9A   0C90          MOVZBL  (R1)+,R0                      ; Get next (final) byte of field
        58   19  07   50   F0   0C93          INSV    R0,#7,#25,R8                  ; Merge with low order bits
                                0C98          $LOG    TPL_PFM,,,R5                  ; Assume fatal event is "format error"
                 16   A5   B7   0CA0          DECW    WQE$L_PM2+2(R5)               ; Account for extra byte
                      4A   19   0CA3          BLSS    40$                           ; If LSS then msg is too small
                                0CA5  10$:    $LOG    TPL_ISF,TPL_PRSN_ADJR,,R5 ; Assume address out of range
        000000FF 8F   58   D1   0CAD          CMPL    R8,#TR2C_MAX_PNA              ; Less than max Phase II address ?
                      31   1A   0CB4          BGTRU   30$                           ; Out of range if GTRU
                      1F   11   0CB6          BRB     20$                           ; Continue in common
                                0CB8
                                0CB8  PARSE_PH3_ADDR:                                 ; Parse Phase III node address field
                                0CB8  PARSE_PH4_ADDR:                                 ; Parse Phase IV node address field
                 58   81   32   0CB8          CVTWL   (R1)+,R8                      ; Get the node address
           03    008A C4   91   0CBB          CMPB    RCB$B_ETY(R4),#ADJ$C_PTY_AREA ; Are we a level 2 router?
                      15   13   0CC0          BEQL    20$                           ; If so, skip the following check
                      0A   EF   0CC2          EXTZV   #TR4$V_ADDR_AREA,-            ; Get the area number
           50   58   06         0CC4                  #TR4$S_ADDR_AREA,R8,R0
                      0E   13   0CC7          BEQL    20$                           ; If area = 0, allow it
           008B C4   50   91   0CC9          CMPB    R0,RCB$B_HOMEAREA(R4)         ; Is it in our area?
                      07   13   0CCE          BEQL    20$                           ; If so, then ok
        12 22 A6     0A   E1   0CD0          BBC     #LPD$V_BC,LPD$W_STS(R6),30$ ; If non-BC circuit, then error
                      26   11   0CD5          BRB     70$                           ; If NI, then simply ignore it so that
                                0CD7                                                ; multiple areas can co-exist without
                                0CD7                                                ; interference from other areas
                      00   EF   0CD7  20$:    EXTZV   #TR4$V_ADDR_DEST,-            ; Extract the node address (from area)
           50   58   0A         0CD9                  #TR4$S_ADDR_DEST,R8,R0
              5A A4   50   B1   0CDC          CMPW    R0,RCB$W_MAX_ADDR(R4)         ; Within bounds?
                      05   1A   0CE0          BGTRU   30$                           ; If not, report error
                 50   01   D0   0CE2          MOVL    #1,R0                         ; Success
```

```
                     0E   11   0CE5  3010          BRB     50$
                                0CE7  3011
                                0CE7  3012 30$:     $LOG    TPL_LDO,-                       ; Line down due to "address out
                                0CE7  3013                  TPL_PRSN_ADJR,,R5               ; of range"
                10 A5     24   90   0CEF  3014 40$:     MOVB    #LEV$C_LOG_ADE,WQE$B_EVT(R5) ; Log event record & shutdown adjacency
                          50   94   0CF3  3015          CLRB    R0                          ; Set error flag
                                0CF5  3016
        00000014'EF       58   B0   0CF5  3017 50$:     MOVW    R8,LEV_W_PNA                ; Save the node address
                          05   0CFC  3018          RSB
                                0CFD  3019
                                0CFD  3020 ;
                                0CFD  3021 ; Drop the message on the floor.
                                0CFD  3022 ;
                                0CFD  3023
                10 A5     00   90   0CFD  3024 70$:     MOVB    #LEV$C_NO_EVT,WQE$B_EVT(R5) ; Do nothing - drop message
                          50   D4   0D01  3025          CLRL    R0                          ; Signal error detected
                          05   0D03  3026          RSB
                                0D04  3027
                                0D04  3028          .DSABL  LSB
```

```
                                0D04  3030          .SBTTL  PARSE_VERSION - Parse version number field
                                0D04  3031 ;+
                                0D04  3032 ; PARSE_VERSION - Parse the 3 byte version number field
                                0D04  3033 ;
                                0D04  3034 ; Inputs:
                                0D04  3035 ;
                                0D04  3036 ;       R6 = LPD address
                                0D04  3037 ;       R5 = WQE address
                                0D04  3038 ;       R4 = RCB address
                                0D04  3039 ;       R1 = Pointer to next byte in msg (version number)
                                0D04  3040 ;
                                0D04  3041 ; Outputs:
                                0D04  3042 ;
                                0D04  3043 ;       R0 = True if Phase IV, else WQE setup to chain to another event
                                0D04  3044 ;       R1 = Advanced past version number field
                                0D04  3045 ;       R8 = First 2 bytes of version number (Version & ECO level)
                                0D04  3046 ;
                                0D04  3047 ;       NET$GL_INITVER = Saved copy of version number (for event logging)
                                0D04  3048 ;
                                0D04  3049 ;       All other registers are preserved.
                                0D04  3050 ;-
                                0D04  3051 PARSE_VERSION:
                     7E   D4    0D04  3052          CLRL    -(SP)                   ; Allocate 4 bytes of scratch space
                                0D06  3053          ;
                                0D06  3054          ;   Get our version number.  This may be one of several values
                                0D06  3055          ;   depending on whether the circuit has been forced to operate
                                0D06  3056          ;   as a certain version type.
                                0D06  3057          ;
               50    1D A6   9A 0D06  3058          MOVZBL  LPD$B_ETY(R6),R0        ; Get our node type for this circuit
             FF 8F    50   91 0D0A  3059          CMPB    R0,#ADJ$C_PTY_UNK       ; Have we been assigned a node type?
                       43   13 0D0E  3060          BEQL    70$                     ; If not, drop msg on the floor
         6E  00000154'EF40   3C 0D10  3061          MOVZWL  PTY_TO_VERSION[R0],(SP) ; Get the 2 byte version number
                                0D18  3062          ;
                                0D18  3063          ;   Compare version numbers.  If we receive a message from a node
                                0D18  3064          ;   with a higher version number, then drop the message.  The
                                0D18  3065          ;   other node will detect that we are lower version and re-send
                                0D18  3066          ;   the correct message.  If the version is lower than ours,
                                0D18  3067          ;   but we don't recognize or support it, then log "version skew".
                                0D18  3068          ;
                     6E   61    91 0D18  3069          CMPB    (R1),(SP)               ; Compare version numbers
                          36    1A 0D1B  3070          BGTRU   70$                     ; If higher than ours, ignore msg
                          07    1F 0D1D  3071          BLSSU   5$                      ; If equal, then
                01 AE     01  A1   91 0D1F  3072          CMPB    1(R1),1(SP)             ; Compare ECO numbers
                          2D    1A 0D24  3073          BGTRU   70$                     ; If higher than ours, ignore msg
                     6E   61    B1 0D26  3074 5$:       CMPW    (R1),(SP)               ; Is it our version?
                          18    12 0D29  3075          BNEQ    60$                     ; If not, version skew
                     58   61    3C 0D2B  3076          MOVZWL  (R1),R8                 ; Return version to caller
         00000000'EF      81    B0 0D2E  3077          MOVW    (R1)+,NET$GL_INITVER    ; Save INIT version (3 bytes)
         00000002'EF      81    90 0D35  3078          MOVB    (R1)+,NET$GL_INITVER+2
                     50   01    D0 0D3C  3079          MOVL    #1,R0                   ; Success
                     5E   04    C0 0D3F  3080 90$:      ADDL    #4,SP                   ; Pop scratch space
                                05 0D42  3081          RSB
                                0D43  3082
                                0D43  3083 ;
                                0D43  3084 ; Version number is lower than ours, but we can't handle it.  Log an event.
                                0D43  3085 ;
                                0D43  3086
```

```
                     0D43  3087 60$:    $LOG    TPL_LDS,TPL_PRSN_VRSK,,R5 ; Setup "version skew" event
10 A5  24  90        0D4B  3088         MOVB    #LEV$C_LOG_ADE,WQE$B_EVT(R5) ; Signal "adjacency down event"
       50  D4        0D4F  3089         CLRL    R0                          ; Signal error detected
       EC  11        0D51  3090         BRB     90$
                     0D53  3091
                     0D53  3092 ;
                     0D53  3093 ; Version number is higher than ours.  Drop the message on the floor.
                     0D53  3094 ;
                     0D53  3095
10 A5  00  90        0D53  3096 70$:    MOVB    #LEV$C_NO_EVT,WQE$B_EVT(R5) ; Do nothing - drop message
       50  D4        0D57  3097         CLRL    R0                          ; Signal error detected
       E4  11        0D59  3098         BRB     90$
```

```
                     0D5B  3100          .SBTTL  SET_DLL_EVT - Schedule event transition
                     0D5B  3101 ;+
                     0D5B  3102 ; SET_DLL_EVT - Schedule the processing of an event
                     0D5B  3103 ;
                     0D5B  3104 ; Inputs:        R0 = Event code
                     0D5B  3105 ;                R6 = LPD address
                     0D5B  3106 ;
                     0D5B  3107 ; Outputs:       R0 = Status
                     0D5B  3108 ;
                     0D5B  3109 ;-
                     0D5B  3110 SET_DLL_EVT::
        07   BB      0D5B  3111          PUSHR   #^M<R0,R1,R2>           ; Save regs
        51   D4      0D5D  3112          CLRL    R1                      ; Indicate no addition WQE space needed
   50   01   D0      0D5F  3113          MOVL    #WQE$C_SUB_ACP,R0       ; Indicate WQE subtype
     F29B'  30      0D62  3114          BSBW    WQE$ALLOCATE            ; Allocate WQE (always succeeds!)
   50   52   D0      0D65  3115          MOVL    R2,R0                   ; Transfer WQE address
10 A0   8E   F6      0D68  3116          CVTLB   (SP)+,WQE$B_EVT(R0)     ; Enter event code
   51   8E   7D      0D6C  3117          MOVQ    (SP)+,R1                ; Recover R1,R2 and cleanup stack
      20 A6   B0      0D6F  3118          MOVW    LPD$W_PTH(R6),-         ; Enter LPD index
      12 A0           0D72  3119                  WQE$W_REQIDT(R0)        ;
      80'AF   9E      0D74  3120          MOVAB   B^NET$DLL_PRC_WQE,-     ; Enter action routine address
      0C A0           0D77  3121                  WQE$L_ACTION(R0)        ;
        F284' 30      0D79  3122          BSBW    WQE$INSQUE              ; Queue the WQE
   50   01   90      0D7C  3123 10$:     MOVB    #1,R0                   ; Indicate success
                     05    0D7F  3124          RSB
```

```
                     0D80  3126          .SBTTL  NET$DLL_PRC_WQE - Process work queue element
                     0D80  3127 ;+
                     0D80  3128 ; NET$DLL_PRC_WQE  - Process Work Queue Element
                     0D80  3129 ;
                     0D80  3130 ; FUNCTIONAL DESCRIPTION:
                     0D80  3131 ;
                     0D80  3132 ; This routine is called by the work queue dispatcher after the WQE is
                     0D80  3133 ; dequeued from the work queue.  The WQE is deallocated below once it has
                     0D80  3134 ; been processed.
                     0D80  3135 ;
                     0D80  3136 ; INPUTS:        R5      WQE address
                     0D80  3137 ;
                     0D80  3138 ;                All other registers are scratch.
                     0D80  3139 ;
                     0D80  3140 ; OUTPUTS:       All registers are clobbered.
                     0D80  3141 ;
                     0D80  3142 ;-
                     0D80  3143 NET$DLL_PRC_WQE:                             ; Process DLL WQE event
      00C7   30      0D80  3144          BSBW    FIND_WQE_CTX                ; Locate CNF, LPD, ADJ blocks
    02 50    E9      0D83  3145          BLBC    R0,KILL_WQE                 ; If LPD no longer exists, skip event
       0B    10      0D86  3146          BSBB    PROC_EVT                    ; Process the event
                     0D88  3147 KILL_WQE:                                    ; Deallocate WQE if its there
 50    55    D0      0D88  3148          MOVL    R5,R0                       ; Get WQE for deallocation
       05    13      0D8B  3149          BEQL    20$                         ; If EQL then none
       55    D4      0D8D  3150          CLRL    R5                          ; Nullify normal pointer to it
     F26E'   30      0D8F  3151          BSBW    WQE$DEALLOCATE              ; Deallocate the WQE
             05      0D92  3152 20$:     RSB                                 ; Done
```

```
                                0D93          .SBTTL  PROC_EVT - Process an event
                                0D93  ;+
                                0D93  ; PROC_EVT      - Process DLL event
                                0D93  ;
                                0D93  ; This routine processes all Data Link Layer events and is state table driven.
                                0D93  ; Action routines are called until the null event is detected.  Each action
                                0D93  ; routine generates a new event, which it returns in R1, and returns with the
                                0D93  ; low bit set in R0 only if the indicated state change is to be performed.
                                0D93  ;
                                0D93  ; Inputs:
                                0D93  ;
                                0D93  ;       R11 = CNR address
                                0D93  ;       R10 = CNF address
                                0D93  ;       R7 = ADJ address
                                0D93  ;       R6 = LPD address
                                0D93  ;       R5 = WQE address
                                0D93  ;
                                0D93  ; Outputs:
                                0D93  ;
                                0D93  ;       R5 = WQE address
                                0D93  ;
                                0D93  ;       All other registers are clobbered
                                0D93  ;-
                                0D93  PROC_EVT:                                 ; Process all DLL events
0000000C'EF    56    D0         0D93          MOVL    R6,LEV_L_LPD              ; Save the LPD pointer
00000010'EF    57    D0         0D9A          MOVL    R7,LEV_L_ADJ              ; Save the ADJ pointer
00000004'EF    5A    7D         0DA1          MOVQ    R10,LEV_Q_CRI             ; Save the CRI CNF and CNR ptrs
                                0DA8          ;
                                0DA8          ;   Find appropriate state table entry
                                0DA8          ;
        51    10 A5    9A       0DA8          MOVZBL  WQE$B_EVT(R5),R1          ; Get the event code
        53    26 A6    9A       0DAC  5$:     MOVZBL  LPD$B_STI(R6),R3          ; Get LPD internal state
           51    24    D1       0DB0          CMPL    S^#LEV$C_MAX_EVT,R1       ; Is event within range ?
                 5C    1F       0DB3          BLSSU   30$                       ; If LSSU then bug exists
     54    51    10    C5       0DB5          MULL3   S^#LEV$C_STATES,R1,R4     ; Bias for current event
           54    53    C0       0DB9          ADDL    R3,R4                     ; Add current state offset
53   00000000'EF44    3E       0DBC          MOVAW   LEV$AW_STA_TAB[R4],R3     ; Address state table entry
                F239'   30       0DC4          BSBW    NET$JNX_CO                ; Initialize journalling co-routine
                                0DC7                                            ; (Clobbers R0; stack has been changed)
              05 50    E9       0DC7          BLBC    R0,14$                    ; If LBC journalling is inactive
                0048   30       0DCA          BSBW    FILL_JNL                  ; Fill the record
                  9E   16       0DCD          JSB     @(SP)+                    ; Store the journal record
           51    01    D1       0DCF  14$:    CMPL    S^#LEV$C_EXIT,R1          ; Are we done ?
                 39    13       0DD2          BEQL    18$                       ; If so, exit processing
                                0DD4          ;
                                0DD4          ;
                                0DD4          ;   Dispatch to the action routine with the following:
                                0DD4          ;
                                0DD4          ;   INPUTS:     R11     CRI CNR ptr
                                0DD4          ;               R10     CRI CNF ptr
                                0DD4          ;               R7      ADJ address
                                0DD4          ;               R6      LPD address
                                0DD4          ;               R5      WQE address
                                0DD4          ;               R4      RCB address
                                0DD4          ;
                                0DD4          ;   ON RETURN:  R5      Unchanged
                                0DD4          ;               R1      Next event to be processed
```

```
                                 0DD4  3211          ;               R0      Low bit set if state change is permitted,
                                 0DD4  3212          ;                       Low bit clear to avoid state change
                                 0DD4  3213          ;
                                 0DD4  3214          ;               All other regs may be clobbered
                                 0DD4  3215          ;
                      83    9F   0DD4  3216          PUSHAB  (R3)+                   ; Save table address
                53    63    9A   0DD6  3217          MOVZBL  (R3),R3                 ; Get action routine index
          50  00000000'EF43    D0   0DD9  3218          MOVL    LEV_AL_ACTTAB[R3],R0    ; Get action routine address
            54  00000000'EF    D0   0DE1  3219          MOVL    NET$GL_PTR_VCB,R4       ; Get RCB pointer
                      60    16   0DE8  3220          JSB     (R0)                    ; Dispatch
                      53  8ED0   0DEA  3221          POPL    R3                      ; Get next state, cleanup stack
            56  0000000C'EF    D0   0DED  3222          MOVL    LEV_L_LPD,R6            ; Get LPD address
                      1A    13   0DF4  3223          BEQL    20$                     ; If EQL then its been deallocated
                   04  50    E9   0DF6  3224          BLBC    R0,15$                  ; Avoid state change if LBC
                26  A6    63    90   0DF9  3225          MOVB    (R3),LPD$B_STI(R6)      ; Change state
            57  00000010'EF    D0   0DFD  3226  15$:    MOVL    LEV_L_ADJ,R7            ; Recover ADJ address
            5A  00000004'EF    7D   0E04  3227          MOVQ    LEV_Q_CRI,R10           ; Recover CRI context
                      9F    11   0E0B  3228          BRB     5$                      ; Process next event
                            0E0D  3229
                    F614    30   0E0D  3230  18$:    BSBW    COND_DEAL_LPD           ; Clean-up if necessary
                      05    0E10  3231  20$:    RSB
                            0E11  3232
                            0E11  3233  30$:    BUG_CHECK  NETNOSTATE,FATAL     ; Signal the bug
                            0E15  3234
                            0E15  3235
                            0E15  3236  FILL_JNL:
                81    CC  8F    90   0E15  3237          MOVB    #^X<CC>,(R1)+           ; Enter journal record type
                81    26  A6    90   0E19  3238          MOVB    LPD$B_STI(R6),(R1)+     ; Current state
              00000000'EF    9F   0E1D  3239          PUSHAB  LEV$AQ_STA_TAB          ; Base of state table
            61    53    8E    C3   0E23  3240          SUBL3   (SP)+,R3,(R1)           ; State table offset
                81    1F    CA   0E27  3241          BICL    #^X1F,(R1)+             ; Display only base offset of event
                81    63    B0   0E2A  3242          MOVW    (R3),(R1)+              ; Enter table contents
              81    20  A6    B0   0E2D  3243          MOVW    LPD$W_PTH(R6),(R1)+     ; Enter path ID
              81    22  A6    B0   0E31  3244          MOVW    LPD$W_STS(R6),(R1)+     ; Enter current status
              81    24  A6    90   0E35  3245          MOVB    LPD$B_XMTFLG(R6),(R1)+  ; Enter xmit flags
              81    1B  A6    90   0E39  3246          MOVB    LPD$B_ASTCNT(R6),(R1)+  ; Enter asynch activity count
              81    04  A7    B0   0E3D  3247          MOVW    ADJ$W_PNA(R7),(R1)+     ; Enter the partner's address
              81    06  A7    B0   0E41  3248          MOVW    ADJ$W_BUFSIZ(R7),(R1)+  ; Enter partner's block size
              81    25  A6    90   0E45  3249          MOVB    LPD$B_PVCFLG(R6),(R1)+  ; Enter PVC startup flags
                      05    0E49  3250          RSB
```

```
                        0E4A  3252          .SBTTL  FIND_WQE_CTX - Find context for a new WQE
                        0E4A  3253 ;+
                        0E4A  3254 ; FIND_WQE_CTX - Find CNF, LPD and ADJ for a new WQE to be processed
                        0E4A  3255 ;
                        0E4A  3256 ; This routine is called to locate the CNF, LPD and ADJ corresponding to a
                        0E4A  3257 ; WQE to be processed.
                        0E4A  3258 ;
                        0E4A  3259 ; Inputs:
                        0E4A  3260 ;
                        0E4A  3261 ;       R5 = WQE address
                        0E4A  3262 ;
                        0E4A  3263 ; Outputs:
                        0E4A  3264 ;
                        0E4A  3265 ;       R11 = CNR address
                        0E4A  3266 ;       R10 = CNF address
                        0E4A  3267 ;       R7 = ADJ address
                        0E4A  3268 ;       R6 = LPD address
                        0E4A  3269 ;       R5 = WQE address
                        0E4A  3270 ;       R0 = True if LPD found, else false
                        0E4A  3271 ;
                        0E4A  3272 ;       R8 is destroyed.
                        0E4A  3273 ;-
                        0E4A  3274 FIND_WQE_CTX:
          00    DD      0E4A  3275          PUSHL   #0                      ; Scratch for holding LPD address
    58    12 A5   3C    0E4C  3276          MOVZWL  WQE$W_REQIDT(R5),R8     ; Get LPD index
                        0E50  3277          ;
                        0E50  3278          ;   If this is the local LPD, then skip looking up the CNF block,
                        0E50  3279          ;   since there is none.
                        0E50  3280          ;
       01   58    91    0E50  3281          CMPB    R8,#LPD$C_LOC_INX       ; Local LPD index?
            2F    13    0E53  3282          BEQL    50$                     ; If so, handle it specially
                        0E55  3283          ;
                        0E55  3284          ;   Find the LPD and CRI
                        0E55  3285          ;
          1DF0    30    0E55  3286          BSBW    NET$GET_LPD_CRI         ; Find LPD and CRI for this index
        25 50     E9    0E58  3287          BLBC    R0,90$                  ; Branch if not found
    6E      56    D0    0E5B  3288          MOVL    R6,(SP)                 ; Save LPD address
                        0E5E  3289          ;
                        0E5E  3290          ;   Find the ADJ.  If the ADJ index in the WQE is zero, then use
                        0E5E  3291          ;   the LPD index so that the static ADJ block is used.
                        0E5E  3292          ;
    58  20 A5    3C    0E5E  3293          MOVZWL  WQE$W_ADJ_INX(R5),R8    ; Is ADJ index 0?
            08    13    0E62  3294          BEQL    10$                     ; Branch if so
          1E5F    30    0E64  3295          BSBW    NET$FIND_ADJ            ; Locate ADJ block
                        0E67  3296                                          ; and get LPD corresponding to ADJ
        16 50     E9    0E67  3297          BLBC    R0,90$                  ; If it went away, skip event
            0C    11    0E6A  3298          BRB     20$
20 A5   20 A6     9B    0E6C  3299 10$:     MOVZBW  LPD$B_PTH_INX(R6),WQE$W_ADJ_INX(R5) ; Set ADJ index
    58  20 A5    3C    0E71  3300          MOVZWL  WQE$W_ADJ_INX(R5),R8    ; Get ADJ index
          1E4E    30    0E75  3301          BSBW    NET$FIND_ADJ            ; Locate ADJ block (returns 0 if none)
                        0E78  3302                                          ; and get LPD corresponding to ADJ
    6E      56    D1    0E78  3303 20$:     CMPL    R6,(SP)                 ; Was ADJ$W_LPD different than WQE LPD?
            0D    12    0E7B  3304          BNEQ    30$                     ; If so, bugcheck
    50      01    D0    0E7D  3305          MOVL    #1,R0                   ; Success
    5E      04    C0    0E80  3306 90$:     ADDL    #4,SP                   ; Pop scratch storage
                  05    0E83  3307          RSB
                        0E84  3308
```

```
                        0E84  3309 ;
                        0E84  3310 ; Special handling for local LPD
                        0E84  3311 ;
                        0E84  3312
 1E19   30              0E84  3313 50$:    BSBW    NET$FIND_LPD            ; Find the LPD via index in R8
F6 50   E8              0E87  3314         BLBS    R0,90$                  ; Exit if ok
                        0E8A  3315                                         ; Else, bugcheck
                        0E8A  3316
                        0E8A  3317 30$:    BUG_CHECK  NETNOSTATE,FATAL     ; Signal the bug
```

```
                     0E8E  3319          .SBTTL  Simple transition routines
                     0E8E  3320 ;+
                     0E8E  3321 ; Various "simple" data link state transition action routines.
                     0E8E  3322 ;
                     0E8E  3323 ; ACT_BUG        - Bugcheck failure
                     0E8E  3324 ; ACT_NYI        - Not-yet-implemented error
                     0E8E  3325 ; ACT_NOP        - No-operation
                     0E8E  3326 ; ACT_IRP_EVT    - I/O Request Packet returned to ACP
                     0E8E  3327 ; ACT_LOG_NFE    - Log event record
                     0E8E  3328 ; ACT_LOG_CDE    - Log event record & shutdown circuit
                     0E8E  3329 ; ACT_LOG_ADE    - Log event record & shutdown adjacency
                     0E8E  3330 ;
                     0E8E  3331 ;    INPUTS:     R11     CRI CNR ptr
                     0E8E  3332 ;                R10     CRI CNF ptr
                     0E8E  3333 ;                R6      LPD address
                     0E8E  3334 ;                R5      WQE address
                     0E8E  3335 ;                R4      RCB address
                     0E8E  3336 ;
                     0E8E  3337 ;    OUTPUTS:    R5      Unchanged
                     0E8E  3338 ;                R1      Next event to be processed
                     0E8E  3339 ;                R0      Low bit set if state change is permitted,
                     0E8E  3340 ;                        Low bit clear to avoid state change
                     0E8E  3341 ;
                     0E8E  3342 ;                All other regs may be clobbered
                     0E8E  3343 ;
                     0E8E  3344 ACT_BUG:
                     0E8E  3345          BUG_CHECK  NETNOSTATE,FATAL     ; Signal the bug
                     0E92  3346 ACT_NYI:
                     0E92  3347          BUG_CHECK  NETNOSTATE,FATAL     ; Signal the bug
                     0E96  3348
                     0E96  3349 ACT_EXIT:                                ; Exit state table processing
51   01   D0           0E96  3350          MOVL    #LEV$C_EXIT,R1         ; Signal last event
50   01   90           0E99  3351          MOVB    #1,R0                  ; Allow state transition
                  05    0E9C  3352          RSB
                     0E9D  3353
                     0E9D  3354 ACT_NOP:                                 ; Nop action routine
51   00   D0           0E9D  3355          MOVL    #LEV$C_NO_EVT,R1       ; Signal last event
50   01   90           0EA0  3356          MOVB    #1,R0                  ; Allow state transition
                  05    0EA3  3357          RSB
                     0EA4  3358
                     0EA4  3359 ACT_LOG_CDE:                             ; Log event record & shutdown circuit
   F159'   30           0EA4  3360          BSBW    NET$EVT_INTRAW         ; Call internal raw event logger
51   11   D0           0EA7  3361          MOVL    #LEV$C_CIN_DOWN,R1     ; Generate circuit down event
50   01   90           0EAA  3362          MOVB    #1,R0                  ; Allow state change
                  05    0EAD  3363          RSB
                     0EAE  3364
                     0EAE  3365 ACT_LOG_ADE:                             ; Log event record & shutdown adjacency
   F14F'   30           0EAE  3366          BSBW    NET$EVT_INTRAW         ; Call internal raw event logger
51   12   D0           0EB1  3367          MOVL    #LEV$C_ADJ_DOWN,R1     ; Generate circuit down event
50   01   90           0EB4  3368          MOVB    #1,R0                  ; Allow state change
                  05    0EB7  3369          RSB
                     0EB8  3370
                     0EB8  3371 ACT_LOG_NFE:                             ; Log event record
   F145'   30           0EB8  3372          BSBW    NET$EVT_INTRAW         ; Call internal raw event logger
51   00   D0           0EBB  3373          MOVL    #LEV$C_NO_EVT,R1       ; No new events
50   01   90           0EBE  3374          MOVB    #1,R0                  ; Allow state change
                  05    0EC1  3375          RSB
```

```
                        0EC2  3377          .SBTTL  ACT_RCV_STR - Received start message
                        0EC2  3378 ;+
                        0EC2  3379 ; ACT_RCV_2STR - Second start message received
                        0EC2  3380 ; ACT_RCV_STR - Start message received
                        0EC2  3381 ;
                        0EC2  3382 ; Inputs:        R11 = CRI CNR address
                        0EC2  3383 ;                R10 = CRI CNF address
                        0EC2  3384 ;                R7 = ADJ address
                        0EC2  3385 ;                R6 = LPD address
                        0EC2  3386 ;                R5 = WQE address
                        0EC2  3387 ;                R4 = RCB address
                        0EC2  3388 ;
                        0EC2  3389 ; Outputs:       R0 = True if state change requested
                        0EC2  3390 ;                R1 = Next event to be processed
                        0EC2  3391 ;
                        0EC2  3392 ;                R6 is the only register preserved.
                        0EC2  3393 ;-
                        0EC2  3394 ACT_RCV_2STR:                            ; Second start msg received
                     88 0EC2  3395          BISB    #LPD$M_XMT_STR!-        ; Retransmit everything
                        0EC3  3396                   LPD$M_XMT_VRF!-
                        0EC3  3397                   LPD$M_XMT_IDLE,-
              24 A6   0E        0EC3  3398                   LPD$B_XMTFLG(R6)
                        0EC6  3399
                        0EC6  3400 ACT_RCV_STR:
                        0EC6  3401          ;
                        0EC6  3402          ;   If the message we just received doesn't match the type
                        0EC6  3403          ;   that we sent, then we need to retransmit a start message
                        0EC6  3404          ;   again, only this time in the right Phase.
                        0EC6  3405          ;
          52  0000014A'EF   9E  0EC6  3406          MOVAB   PTY_TO_PHASE,R2         ; Get address of phase conversion table
               50   1D A6   9A  0ECD  3407          MOVZBL  LPD$B_ETY(R6),R0        ; Get our node type
          51  00000038'EF   9A  0ED1  3408          MOVZBL  PTYPE,R1                ; Get his node type (from msg)
              6240   6241   91  0ED8  3409          CMPB    (R2)[R1],(R2)[R0]       ; Does msg "phase" match ours?
                       25   13  0EDD  3410          BEQL    5$                      ; If same phase, process message
                       1E   14  0EDF  3411          BGTR    4$                      ; If higher phase, ignore msg.
                        0EE1  3412                                                  ; Else, restart init. sequence by ...
                        0EE1  3413          SETBIT  LPD$V_XMT_STR,-                 ; Retransmit start msg (the start
                        0EE1  3414                  LPD$B_XMTFLG(R6)                ; msg we already sent was wrong)
                        0EE5  3415          ;
                        0EE5  3416          ;   The message type we received doesn't match what we sent.
                        0EE5  3417          ;
                        0EE5  3418          ;   If this circuit has been forced into a certain type by
                        0EE5  3419          ;   the network manager, then ignore the message we just
                        0EE5  3420          ;   received because it doesn't match what we want.
                        0EE5  3421          ;   Otherwise, process the message (storing the correct type),
                        0EE5  3422          ;   and we will later send the correct start message because
                        0EE5  3423          ;   we were just marked for "start transmit" above.
                        0EE5  3424          ;
                     57   DD  0EE5  3425          PUSHL   R7                      ; Save ADJ address
                        0EE7  3426          $GETFLD cri,l,xpt                   ; Were we forced into a specific type?
                     57 8ED0  0EF4  3427          POPL    R7                      ; Restore ADJ address
                  05 50   E8  0EF7  3428          BLBS    R0,4$                   ; If so, then ignore the message
                   00B0   30  0EFA  3429          BSBW    ADAPT_TO_PARTNER        ; Adapt to partner's node type
                     05   11  0EFD  3430          BRB     5$                      ; Process the message now
                        0EFF  3431          ;
                        0EFF  3432          ;   Ignore the message
                        0EFF  3433          ;
```

```
                  50   94  0EFF  4$:     CLRB    R0                          ; No state change
                0096   31  0F01          BRW     40$
                           0F04          ;
                           0F04          ;   Process message.  Dispatch on message type.
                           0F04          ;
                           0F04  5$:     $DISPATCH PTYPE,<-                  ; Dispatch on message type
                           0F04                  <ADJ$C_PTY_PH2,7$>,-        ; Phase II
                           0F04                  <ADJ$C_PTY_PH3,8$>>         ; Phase III routing
                  19   11  0F12          BRB     10$                         ; Else, no special processing here
                           0F14          ;
                           0F14          ;   Partner is a Phase II node.  Set the number of NOP messages to
                           0F14          ;   send for circuit acceptance test.
                           0F14          ;
                  00   90  0F14  7$:     MOVB    #TR2C_NUM_NOP,-             ; Setup # of NOP msgs to send
               1A A6       0F16                  LPD$B_TSTCNT(R6)            ; to test the circuit
                  17   11  0F18          BRB     20$
                           0F1A          ;
                           0F1A          ;   Partner is a Phase III routing node.  Determine if his receive
                           0F1A          ;   buffer size is adequate to receive a maximum sized routing message.
                           0F1A          ;
            50   5A A4  3C  0F1A  8$:     MOVZWL  RCB$W_MAX_ADDR(R4),R0       ; Get max node address
               50   02  C4  0F1E          MULL    #NET$C_TRCTL_CEL,R0         ; Cell size for node in
                           0F21                                              ; routing message
               50   05  C0  0F21          ADDL    #NET$C_TRCTL_OVR,R0         ; Routing message overhead
      50   00000018'EF  B1  0F24          CMPW    LEV_W_BLKSIZE,R0            ; Can partner receive rtg msg?
                  71   1F  0F2B          BLSSU   50$                         ; If LSSU buffer is too small
                           0F2D          ;
                           0F2D          ;   Setup number of test messages to send for circuit acceptance test
                           0F2D          ;   for all nodes except Phase II nodes.
                           0F2D          ;
                  03   90  0F2D  10$:    MOVB    #TR3C_NUM_TST,-             ; Setup number of test messages
               1A A6       0F2F                  LPD$B_TSTCNT(R6)
                           0F31          ;
                           0F31          ;   Store partner's block size, set flags to schedule initialization
                           0F31          ;   message transmission.
                           0F31          ;
  00C0 8F   00000018'EF  B1  0F31  20$:    CMPW    LEV_W_BLKSIZE,#NET$C_MINBUFSIZ  ; At least as big as minimum?
                  62   1F  0F3A          BLSSU   50$                             ; If LSSU then no
     06 A7   00000018'EF  B0  0F3C          MOVW    LEV_W_BLKSIZE,ADJ$W_BUFSIZ(R7)  ; Setup partner's buff size
     04 A7   00000014'EF  B0  0F44          MOVW    LEV_W_PNA,ADJ$W_PNA(R7)         ; Setup partner's node address
        50   00000020'EF  3C  0F4C          MOVZWL  LEV_W_HELLO,R0                  ; Get partner's hello timer
                  04   12  0F53          BNEQ    22$                             ; If not specified (Phase III),
            50   18 A6  3C  0F55          MOVZWL  LPD$W_INT_TLK(R6),R0            ; then use our own hello timer
               50   02  C4  0F59  22$:    MULL    #TR4C_T3MULT,R0                 ; Multiply by hello/listen factor
            08 A7   50  B0  0F5C          MOVW    R0,ADJ$W_INT_LSN(R7)            ; Set listen interval
            0A A7   50  B0  0F60          MOVW    R0,ADJ$W_TIM_LSN(R7)            ; Start listen timer
     01 A7   00000038'EF  90  0F64          MOVB    PTYPE,ADJ$B_PTYPE(R7)           ; Setup partner's node type
                           0F6C          $DISPATCH PTYPE,<-
                           0F6C                  <ADJ$C_PTY_PH3,25$>,-           ; If Phase III router,
                           0F6C                  <ADJ$C_PTY_PH4,25$>,-           ; Or Phase IV level 1 router,
                           0F6C                  <ADJ$C_PTY_AREA,25$>>           ; Or Phase IV level 2 router,
                  03   11  0F7E          BRB     28$
                           0F80  25$:    SETBIT  ADJ$V_RTG,ADJ$B_STS(R7)         ; then set RTG flag
     24 A6   00000034'EF  88  0F83  28$:    BISB    XMTFLG,LPD$B_XMTFLG(R6)         ; Setup xmit flags
     04 00000034'EF  02  E0  0F8B          BBS     #LPD$V_XMT_VRF,XMTFLG,30$       ; Br if verification msg needed
                           0F93          CLRBIT  LPD$V_XMT_VRF,LPD$B_XMTFLG(R6)  ; Clear flag to send the msg
               50   01  90  0F97  30$:    MOVB    #1,R0                           ; Allow state change
```

```
                  0F9A
51   00   D0      0F9A  40$:    MOVL    S^#LEV$C_NO_EVT,R1              ; No further events
          05      0F9D          RSB
                  0F9E
                  0F9E  ;
                  0F9E  ; Log "invalid partner block size" event
                  0F9E  ;
                  0F9E  50$:    $LOG    TPL_IOF,TPL_PRSN_ADJB,,R5       ; Buffer size too small
51   23   D0      0FA6          MOVL    #LEV$C_LOG_CDE,R1               ; Signal "circuit down event"
50   01   D0      0FA9          MOVL    #1,R0                           ; Make state change
          05      0FAC          RSB
```

```
                                                C 9
                     0FAD  3503          .SBTTL  ADAPT_TO_PARTNER - Adapt to partner's node type
                     0FAD  3504 ;+
                     0FAD  3505 ; ADAPT_TO_PARTNER - Adapt to partner's node type
                     0FAD  3506 ;
                     0FAD  3507 ; This routine is called when our partner is speaking a older Routing
                     0FAD  3508 ; version than we are.  The function is to figure out what version we want
                     0FAD  3509 ; to speak, based on his version, and store it in the LPD block so that
                     0FAD  3510 ; we only speak the older version from now on.
                     0FAD  3511 ;
                     0FAD  3512 ; Inputs:
                     0FAD  3513 ;
                     0FAD  3514 ;       R7 = ADJ address
                     0FAD  3515 ;       R6 = LPD address
                     0FAD  3516 ;       R4 = RCB address
                     0FAD  3517 ;       PTYPE = Partner node type (based on Start message)
                     0FAD  3518 ;       LPD$B_ETY = Our node type
                     0FAD  3519 ;
                     0FAD  3520 ; Outputs:
                     0FAD  3521 ;
                     0FAD  3522 ;       LPD$B_ETY = Our new adapted node type for this circuit
                     0FAD  3523 ;
                     0FAD  3524 ;       R0-R2 are destroyed.
                     0FAD  3525 ;-
                     0FAD  3526 ADAPT_TO_PARTNER:
   51  00000038'EF   9A  0FAD  3527          MOVZBL  PTYPE,R1                ; Get partner node type
    50   1D A6      9A  0FB4  3528          MOVZBL  LPD$B_ETY(R6),R0        ; Get our node type
                     0FB8  3529          $DISPATCH R0,<-                 ; If we are an endnode,
                     0FB8  3530                  <ADJ$C_PTY_PH4N,20$>,-  ; then drop to his version only as
                     0FB8  3531                  <ADJ$C_PTY_PH3N,20$>>   ; an endnode
                     0FC6  3532          ;
                     0FC6  3533          ;   We are a router.  Make sure that we drop to his version,
                     0FC6  3534          ;   BUT as a router of that version, and not an endnode.
                     0FC6  3535          ;
   52  0000014A'EF41   9A  0FC6  3536          MOVZBL  PTY_TO_PHASE[R1],R2     ; Get his "phase" (II, III or IV)
                     0FCE  3537          $DISPATCH R2,<=                 ; Dispatch on phase #
                     0FCE  3538                  <2,50$>,-               ; Drop to Phase II
                     0FCE  3539                  <3,13$>>                ; Drop to Phase III router
         1F  11      0FD6  3540          BRB     70$                     ; All other values illegal
                     0FD8  3541
      51  00  90     0FD8  3542 13$:     MOVB    #ADJ$C_PTY_PH3,R1       ; Act as a Phase III router
         15  11      0FDB  3543          BRB     50$
                     0FDD  3544
                     0FDD  3545          ;
                     0FDD  3546          ;   We are an endnode.  Make sure that we drop to his version,
                     0FDD  3547          ;   BUT as an endnode of that version, and not a router.
                     0FDD  3548          ;
   52  0000014A'EF41   9A  0FDD  3549 20$:     MOVZBL  PTY_TO_PHASE[R1],R2     ; Get his "phase" (II, III or IV)
                     0FE5  3550          $DISPATCH R2,<=                 ; Dispatch on phase #
                     0FE5  3551                  <2,50$>,-               ; Phase II
                     0FE5  3552                  <3,23$>>                ; Phase III endnode
         08  11      0FED  3553          BRB     70$                     ; All other values illegal
                     0FEF  3554
      51  01  90     0FEF  3555 23$:     MOVB    #ADJ$C_PTY_PH3N,R1      ; Act as a Phase III endnode
                     0FF2  3556
    1D A6  51  90    0FF2  3557 50$:     MOVB    R1,LPD$B_ETY(R6)        ; Set our new node type
                 05  0FF6  3558          RSB
                     0FF7  3559
```

```
;
; Cannot adapt to his version
;
70$:    BUG_CHECK NETNOSTATE,FATAL
```

```
                        0FFB  3566          .SBTTL  ACT_RCV_VRF - Received verification message
                        0FFB  3567 ;+
                        0FFB  3568 ; ACT_RCV_VRF - React to received verification message
                        0FFB  3569 ;
                        0FFB  3570 ; Inputs:        R11 = CRI CNR address
                        0FFB  3571 ;                R10 = CRI CNF address
                        0FFB  3572 ;                R7 = ADJ address
                        0FFB  3573 ;                R6 = LPD address
                        0FFB  3574 ;                R5 = WQE address
                        0FFB  3575 ;
                        0FFB  3576 ; Outputs:       R0 = True if state change requested
                        0FFB  3577 ;                R1 = Next event to be processed
                        0FFB  3578 ;
                        0FFB  3579 ;                R6 is the only register preserved.
                        0FFB  3580 ;-
                        0FFB  3581 ACT_RCV_VRF:
                        0FFB  3582          ;
                        0FFB  3583          ;   Is phase of verification message the same as that of the Init msg?
                        0FFB  3584          ;
         58   04 A7  3C  0FFB  3585          MOVZWL   ADJ$W_PNA(R7),R8       ; Get partner's address
                        0FFF  3586          $LOG    TPL_ISF,TPL_PRSN_UXPK,,R5 ; Assume phase change
                        1007  3587
         02   01 A7  91  1007  3588          CMPB    ADJ$B_PTYPE(R7),#ADJ$C_PTY_PH2 ; Phase II message expected?
                   0C  12  100B  3589          BNEQ    5$                      ; Branch if not
   02    00000038'EF  91  100D  3590          CMPB    PTYPE,#ADJ$C_PTY_PH2    ; Phase II message?
                   1D  13  1014  3591          BEQL    10$                     ; Ok if so
                 00B9  31  1016  3592 4$:      BRW     30$                     ; Else phase change - log it
   02    00000038'EF  91  1019  3593 5$:      CMPB    PTYPE,#ADJ$C_PTY_PH2    ; Phase II message?
                   F4  13  1020  3594          BEQL    4$                      ; If so, phase change - log it
                       1022  3595          ;
                       1022  3596          ;   Did the operator change the adjacent node's address?
                       1022  3597          ;
                       1022  3598          $LOG    TPL_IOF,TPL_PRSN_ADJC,,R5 ; Assume address change
00000014'EF    58   B1  102A  3599          CMPW    R8,CEV_W_PNA            ; Is the address the same as it was?
                   E3  12  1031  3600          BNEQ    4$                      ; If not the same, log the event
                       1033  3601          ;
                       1033  3602          ;   If the remote node is a Phase III router, then REQUIRE that
                       1033  3603          ;   a non-null password has been specified on the remote node,
                       1033  3604          ;   and that it matches the receive password specified on this node.
                       1033  3605          ;   This is intended to prevent accidental "cost/hops leakage"
                       1033  3606          ;   (i.e. address merging) between two different areas if they
                       1033  3607          ;   are accidentally connected between 1 or more Phase III routers.
                       1033  3608          ;   The recommendation to customers is that they use passwords
                       1033  3609          ;   containing the area number so that Phase III node can't be
                       1033  3610          ;   accidentally connected to another area.
                       1033  3611          ;
                       1033  3612          ;   This check is only done if we are in a hierarchical network,
                       1033  3613          ;   which is assumed if our homearea is not "1".
                       1033  3614          ;
       0C80  8F    BB  1033  3615 10$:     PUSHR   #^M<R7,R10,R11>         ; Save registers
             53    D4  1037  3616          CLRL    R3                      ; Preset "exact match" flag = false
   01  008B  C4  91  1039  3617          CMPB    RCB$B_HOMEAREA(R4),#1   ; Is our area anything but "1"?
                   2A  13  103E  3618          BEQL    11$                     ; Branch if not
         01    A7  91  1040  3619          CMPB    ADJ$B_PTYPE(R7),-       ; Is adjacent node Phase III router?
                   00  1043  3620                  #ADJ$C_PTY_PH3
                   24  12  1044  3621          BNEQ    11$                     ; Branch if not
             53    01  D0  1046  3622          MOVL    #1,R3                   ; Require exact password match
```

```
                08   BB  1049  3623          PUSHR   #^M<R3>                         ; Save flag
          00    6E   00   2D  104B  3624          CMPC5   #0,(SP),#0,-                    ; Check if remote node gave
            00000024'EF             104F  3625                  LEV_Q_PSWDESC,-             ; null password (RSX sends 8 bytes
            00000028'FF             1054  3626                  @LEV_Q_PSWDESC+4            ; of 0, rather than 0 byte string)
                08   BA  1059  3627          POPR    #^M<R3>                         ; Restore flag - use POPR, saving PSL
                1D   12  105B  3628          BNEQ    12$                             ; If null password,
                         105D  3629          $LOG    TPL_ISF,TPL_PRSN_VREQ,,R5       ; Log "verification req. from PH3 node"
             18 A5   D4  1065  3630          CLRL    WQE$L_EVL_PRT(R5)               ; Don't print any packet header either
                64   11  1068  3631          BRB     29$                             ; log event and bring down circuit
                         106A  3632          ;
                         106A  3633          ;   If the circuit has specified that verification is required
                         106A  3634          ;   for all remote nodes over this circuit, then require verification.
                         106A  3635          ;
                         106A  3636  11$:    $GETFLD cri,v,ver                       ; Is verification required ?
                         1077  3637          ASSUME  NMA$C_CIRVE_ENA  EQ  0
                         1077  3638          ASSUME  NMA$C_CIRVE_DIS  EQ  1
             49 58   E8  1077  3639          BLBS    R8,20$                          ; If disabled (or not set), skip it
                         107A  3640                                                  ; Require match only if RPA set
                         107A  3641          ;
                         107A  3642          ;   Verification is required.  Does the receive password match?
                         107A  3643          ;   If no node database entry is found, or if the passwords don't
                         107A  3644          ;   match, then reject the connection.
                         107A  3645          ;
    5B   00000000'EF     D0  107A  3646  12$:    MOVL    NET$GL_CNR_NDI,R11              ; Setup the root pointer
    58   00000014'EF     3C  1081  3647          MOVZWL  LEV_W_PNA,R8                    ; Get node address
               EF75'     30  1088  3648          BSBW    NET$NDI_BY_ADD                  ; Find the matching NDI
             0A 50   E8  108B  3649          BLBS    R0,15$                          ; If LBS then found
                         108E  3650          $LOG    TPL_VFR,,,R5                    ; due to "node not in database"
                36   11  1096  3651          BRB     29$                             ; Log the event and bring line down
                         1098  3652  15$:    $GETFLD ndi,s,rpa                       ; Get the receive password
             03 53   E8  10A5  3653          BLBS    R3,18$                          ; If exact match not required,
             18 50   E9  10A8  3654          BLBC    R0,20$                          ; and no password specified, skip check
                         10AB  3655                                                  ; (else, try match even with null RPA)
                         10AB  3656  18$:    $LOG    TPL_VFR,,,R5                    ; Assume password mismatch
          00    68   57   2D  10B3  3657          CMPC5   R7,(R8),#0,-                    ;
            00000024'EF             10B7  3658                  LEV_Q_PSWDESC,-             ; Does it match ?
            00000028'FF             10BC  3659                  @LEV_Q_PSWDESC+4            ;
                0B   12  10C1  3660          BNEQ    29$                             ; If NEQ no - verification failure
           0C80 8F   BA  10C3  3661  20$:    POPR    #^M<R7,R10,R11>                 ; Restore registers
             51 0A   D0  10C7  3662          MOVL    S^#LEV$C_RCV_VVF,R1             ; Indicate "valid verification"
             50 01   90  10CA  3663          MOVB    #1,R0                           ; Allow state change
                05   10CD  3664          RSB
                         10CE  3665
           0C80 8F   BA  10CE  3666  29$:    POPR    #^M<R7,R10,R11>                 ; Restore registers
                         10D2  3667                                                  ; and log verification failure
                         10D2  3668  ;
                         10D2  3669  ; Log verification failure and bring the line down
                         10D2  3670  ;
             51 23   D0  10D2  3671  30$:    MOVL    #LEV$C_LOG_CDE,R1               ; Switch to circuit down event
             50 01   D0  10D5  3672          MOVL    #1,R0                           ; Make state change
                05   10D8  3673          RSB
```

```
                                        10D9  3675          .SBTTL  ACT_RCV_RHEL - Received Router Hello message
                                        10D9  3676 ;+
                                        10D9  3677 ; ACT_RCV_RHEL - Router Hello message received
                                        10D9  3678 ;
                                        10D9  3679 ; Inputs:        R11 = CRI CNR address
                                        10D9  3680 ;                R10 = CRI CNF address
                                        10D9  3681 ;                R7 = ADJ address
                                        10D9  3682 ;                R6 = LPD address
                                        10D9  3683 ;                R5 = WQE address
                                        10D9  3684 ;
                                        10D9  3685 ; Outputs:       R0 = True if state change requested
                                        10D9  3686 ;                R1 = Next event to be processed
                                        10D9  3687 ;
                                        10D9  3688 ;                R6 is the only register preserved.
                                        10D9  3689 ;-
                                        10D9  3690 ACT_RCV_RHEL:
                                        10D9  3691                  ;
                                        10D9  3692                  ;   Check that buffer size is reasonable
                                        10D9  3693                  ;
00C0 8F   00000018'EF     B1            10D9  3694                  CMPW    LEV_W_BLKSIZE,#NET$C_MINBUFSIZ  ; At least as big as minimum?
                    0B    1E            10E2  3695                  BGEQU   10$                             ; If LSSU then no
                                        10E4  3696                  $LOG    TPL_IOF,TPL_PRSN_ADJB,,R5       ; Buffer size too small
                  00F5    31            10EC  3697                  BRW     70$                             ; Log the event, bring adj down
                                        10EF  3698 10$:             ;
                                        10EF  3699                  ;   Check that partner's node type hasn't changed
                                        10EF  3700                  ;
01 A7     00000038'EF     91            10EF  3701                  CMPB    PTYPE,ADJ$B_PTYPE(R7)           ; Node type changed?
                    0B    13            10F7  3702                  BEQL    20$                             ; Branch if ok
                                        10F9  3703                  $LOG    TPL_LDS,TPL_PRSN_UXPK,,R5       ; Unexpected message
                  00E0    31            1101  3704                  BRW     70$                             ; Log the event, bring adj down
                                        1104  3705 20$:             ;
                                        1104  3706                  ;   Store partner's block size, router priority and listen timer
                                        1104  3707                  ;   parsed from message.
                                        1104  3708                  ;
06 A7     00000018'EF     B0            1104  3709                  MOVW    LEV_W_BLKSIZE,ADJ$W_BUFSIZ(R7)  ; Setup partner's buff size
0C A7     0000001C'EF     90            110C  3710                  MOVB    LEV_B_PRIORITY,ADJ$B_BCPRI(R7)  ; Set router priority
   50     00000020'EF     3C            1114  3711                  MOVZWL  LEV_W_HELLO,R0                  ; Get partner's hello timer
             50    03     C4            111B  3712                  MULL    #TR4C_BCT3MULT,R0               ; Multiply by hello/listen factor
          08 A7    50     B0            111E  3713                  MOVW    R0,ADJ$W_INT_LSN(R7)            ; Set listen interval
          0A A7    50     B0            1122  3714                  MOVW    R0,ADJ$W_TIM_LSN(R7)            ; Start listen timer
                                        1126  3715                  ;
                                        1126  3716                  ;   If this router buffer size is less than the current "minimum",
                                        1126  3717                  ;   then we want to update the main ADJ$W_BUFSIZ for the BC, so that
                                        1126  3718                  ;   it always contains the minimum buffer size of all BRAs on the NI.
                                        1126  3719                  ;
          50    20 A6     9A            1126  3720                  MOVZBL  LPD$B_PTH_INX(R6),R0            ; Get LPD index
       50    2C B440     D0            112A  3721                  MOVL    @RCB$L_PTR_ADJ(R4)[R0],R0       ; Get main ADJ for BC
    06 A0     06 A7     B1            112F  3722                  CMPW    ADJ$W_BUFSIZ(R7),ADJ$W_BUFSIZ(R0) ; Is bufsiz less than minimum?
                 05     1E            1134  3723                  BGEQU   25$                             ; Branch if not
    06 A0     06 A7     B0            1136  3724                  MOVW    ADJ$W_BUFSIZ(R7),ADJ$W_BUFSIZ(R0) ; If so, store the minimum
                                        113B  3725 25$:             ;
                                        113B  3726                  ;   If we are an endnode, then simply remember this router as being
                                        113B  3727                  ;   our "designated router", and mark the BRA up.
                                        113B  3728                  ;
                                        113B  3729                  $DISPATCH LPD$B_ETY(R6),TYPE=B,<-       ; If we are an endnode,
                                        113B  3730                          <ADJ$C_PTY_PH4N,27$>,-
                                        113B  3731                          <ADJ$C_PTY_PH3N,27$>>
```

```
                1A   11  114A  3732          BRB     29$
        0B 67   01   E2  114C  3733 27$:     BBSS    #ADJ$V_RUN,ADJ$B_STS(R7),26$     ; Set into RUN state
                         1150  3734          $LOG    TPL_AUP,,,R5                      ; Set "adjacency up" event
            EEA5'   30   1158  3735          BSBW    NET$EVT_INTRAW                    ; Log the event record
    2C A6   20 A5   B0   115B  3736 26$:     MOVW    WQE$W_ADJ_INX(R5),LPD$W_DRT(R6)  ; Store ADJ index of DRT
            EE9D'   30   1160  3737          BSBW    UPDATE_ALL                        ; Update output path if changed
            0064    31   1163  3738          BRW     90$                               ; Exit with success
                         1166  3739 29$:     ;
                         1166  3740          ;   See if two-way communication has been reached with the remote
                         1166  3741          ;   partner by checking to see if our node address appears in his
                         1166  3742          ;   election list.
                         1166  3743          ;
        52   16 A5   3C  1166  3744          MOVZWL  WQE$L_PM2+2(R5),R2                ; Get number of bytes in list
        53   14 A5   3C  116A  3745          MOVZWL  WQE$L_PM2(R5),R3                  ; Get offset to list
          53    55   C0  116E  3746          ADDL    R5,R3                             ; Get address of list
                16   11  1171  3747          BRB     40$
  000400AA 8F   63   D1  1173  3748 30$:     CMPL    (R3),#TR$C_NI_PREFIX              ; Standard Phase IV prefix?
                23   12  117A  3749          BNEQ    69$                               ; If not, packet format error
    0E A4   04 A3   B1  117C  3750          CMPW    4(R3),RCB$W_ADDR(R4)              ; Our address?
                1F   13  1181  3751          BEQL    42$                               ; Branch if so
          52    07   C2  1183  3752 35$:     SUBL    #7,R2                             ; Skip entry in list
          53    07   C0  1186  3753          ADDL    #7,R3
                52   D5  1189  3754 40$:     TSTL    R2                                ; Any bytes left?
                E6   14  118B  3755          BGTR    30$
                         118D  3756          ;
                         118D  3757          ;   Our node address is not in his list.  If this adjacency
                         118D  3758          ;   was already up, the bring it down (since it represents
                         118D  3759          ;   a change in his status).  If we are waiting for 2-way
                         118D  3760          ;   communication, then continue to wait.
                         118D  3761          ;
                01   E1  118D  3762          BBC     #ADJ$V_RUN,-                ; If not in RUN state,
            23 67        118F  3763                  ADJ$B_STS(R7),45$           ; then keep waiting
                         1191  3764          $LOG    TPL_LDS,TPL_PRSN_DROP,,R5 ; Log "dropped by adjacent node"
            18 A5   D4   1199  3765          CLRL    WQE$L_EVL_PKT(R5)           ; Don't print any packet header either
            0045    31   119C  3766          BRW     70$                         ; Bring down adjacency
                         119F  3767
            002F    31   119F  3768 69$:     BRW     60$                         ; Report packet format error
                         11A2  3769          ;
                         11A2  3770          ;   2-way communication has been established to the partner node.
                         11A2  3771          ;   Mark the broadcast router adjacency in run state and log the
                         11A2  3772          ;   adjacency up.
                         11A2  3773          ;
                01   E2  11A2  3774 42$:     BBSS    #ADJ$V_RUN,-                ; Set into RUN state
            0E 67        11A4  3775                  ADJ$B_STS(R7),45$           ; Skip if already in run state
                         11A6  3776          $LOG    TPL_AUP,,,R5                ; Set "adjacency up" event
            EE4F'   30   11AE  3777          BSBW    NET$EVT_INTRAW              ; Log the event record
            EE4C'   30   11B1  3778          BSBW    UPDATE_ALL                  ; Force routing msgs to be sent
                         11B4  3779 45$:     ;
                         11B4  3780          ;   Build new NI router list for our Router Hello message
                         11B4  3781          ;
                         11B4  3782          ;   If there was at least one router in the election list which
                         11B4  3783          ;   is still waiting for two-way communication to be established,
                         11B4  3784          ;   then cause a Router Hello message to be sent in 1 second, so
                         11B4  3785          ;   that elections are resolved quickly on the NI.
                         11B4  3786          ;
            0FE6    30   11B4  3787          BSBW    BUILD_RTR_LIST                    ; Re-build NI router list
            04 50   E8   11B7  3788          BLBS    R0,48$                            ; Branch if election stabilized
```

```
                16 A6   01    B0  11BA  3789                MOVW    #1,LPD$W_TIM_TLK(R6)            ; Make talker fire in 1 sec.
                                  11BE  3790 48$:            ;
                                  11BE  3791                 ;   If we have just initialized the circuit ("waiting for ballots"),
                                  11BE  3792                 ;   then wait to resolve the election after we've had time to hear
                                  11BE  3793                 ;   from everyone.  The election timer will eventually resolve the
                                  11BE  3794                 ;   election in this case.
                                  11BE  3795                 ;
                      0F    E0    11BE  3796                 BBS     #LPD$V_ELECT_TIM,-       ; If we are waiting for ballots,
                  07 22 A6        11C0  3797                         LPD$W_STS(R6),90$        ; then wait for timer to fire
                                  11C3  3798                 ;
                                  11C3  3799                 ;   Store designated router address in LPD.  If we are the
                                  11C3  3800                 ;   designated router, then NETDRIVER will send our Router Hello
                                  11C3  3801                 ;   messages to "all endnodes" as well as "all routers".
                                  11C3  3802                 ;
                    105A  30      11C3  3803                 BSBW    ELECT_ROUTER             ; Elect a designated router
                2C A6   51        11C6  3804                 MOVW    R1,LPD$W_DRT(R6)         ; Store designated router index
                   51   00    D0  11CA  3805 90$:            MOVL    S^#LEV$C_NO_EVT,R1       ; No further events
                   50   01    90  11CD  3806                 MOVB    #1,R0                    ; Allow state change
                             05  11D0  3807                 RSB
                                  11D1  3808
                                  11D1  3809 ;
                                  11D1  3810 ; Log "packet format error" & bring adjacency down
                                  11D1  3811 ;
                                  11D1  3812 60$:            BUMP    B,RCB$B_CNT_PFE(R4)      ; Bump packet format error count
                                  11DC  3813                 $LOG    TPL_PFM,,,R5             ; Packet format error
                                  11E4  3814
                                  11E4  3815 ;
                                  11E4  3816 ; Log event record & bring adjancency down
                                  11E4  3817 ;
                   51   24    D0  11E4  3818 70$:            MOVL    #LEV$C_LOG_ADE,R1        ; Signal "adjacency down event"
                   50   01    D0  11E7  3819                 MOVL    #1,R0                    ; Make state change
                             05  11EA  3820                 RSB
```

```
                                J 9
                                .SBTTL  ACT_ELECT - Resolve election after waiting for ballots
                                ;+
                                ; ACT_ELECT - Resolve election which is waiting for ballots
                                ;
                                ; This routine must only be called if we are a router (if an endnode
                                ; was to set its DRT to ourself, we would probably crash).
                                ;
                                ; Inputs:
                                ;
                                ;       R11 = CRI CNR address
                                ;       R10 = CRI CNF address
                                ;       R7 = ADJ address
                                ;       R6 = LPD address
                                ;       R5 = WQE address
                                ;       R4 = RCB address
                                ;
                                ; Outputs:
                                ;
                                ;       R0 = True if state change requested
                                ;       R1 = Next event to be processed
                                ;
                                ;       R6 is the only register preserved.
                                ;-
                                ACT_ELECT:
                                        ;
                                        ;   Clear the election suppression flag.  This means that after
                                        ;   this point, the routine which receives the Router Hello messages
                                        ;   will be free to run the election algorithm itself.
                                        ;
                                CLRBIT  #LPD$V_ELECT_TIM,-              ; Clear election suppression
                                        LPD$W_STS(R6)
                                ;
                                ;   Store designated router address in LPD.  If we are the
                                ;   designated router, then NETDRIVER will send our Router Hello
                                ;   messages to "all endnodes" as well as "all routers".
                                ;
 102D   30                      BSBW    ELECT_ROUTER                    ; Elect a designated router
2C A6    51   B0                MOVW    R1,LPD$W_DRT(R6)                ; Store designated router index
                                ;
   51    00   D0                MOVL    S^#LEV$C_NO_EVT,R1              ; No further events
   50    01   90                MOVB    #1,R0                           ; Allow state change
              05                RSB
```

```
                                        11FE  3865          .SBTTL  ACT_RCV_EHEL - Received Endnode Hello message
                                        11FE  3866 ;+
                                        11FE  3867 ; ACT_RCV_EHEL - Endnode Hello message received
                                        11FE  3868 ;
                                        11FE  3869 ; Inputs:        R11 = CRI CNR address
                                        11FE  3870 ;                R10 = CRI CNF address
                                        11FE  3871 ;                R7 = ADJ address
                                        11FE  3872 ;                R6 = LPD address
                                        11FE  3873 ;                R5 = WQE address
                                        11FE  3874 ;                R4 = RCB address
                                        11FE  3875 ;
                                        11FE  3876 ; Outputs:       R0 = True if state change requested
                                        11FE  3877 ;                R1 = Next event to be processed
                                        11FE  3878 ;
                                        11FE  3879 ;                R6 is the only register preserved.
                                        11FE  3880 ;-
                                        11FE  3881 ACT_RCV_EHEL:
                                        11FE  3882          ;
                                        11FE  3883          ;   Check that buffer size is reasonable
                                        11FE  3884          ;
00C0 8F   00000018'EF     B1    11FE    3885          CMPW    LEV_W_BLKSIZE,#NET$C_MINBUFSIZ  ; At least as big as minimum?
                    0A    1E    1207    3886          BGEQU   10$                             ; If LSSU then no
                                1209    3887          $LOG    TPL_IOF,TPL_PRSN_ADJB,,R5       ; Buffer size too small
                    35    11    1211    3888          BRB     70$                             ; Log the event, bring adj down
                                1213    3889 10$:       ;
                                1213    3890          ;   Check that partner's node type hasn't changed
                                1213    3891          ;
 01 A7    00000038'EF     91    1213    3892          CMPB    PTYPE,ADJ$B_PTYPE(R7)           ; Node type changed?
                    0A    13    121B    3893          BEQL    20$                             ; Branch if ok
                                121D    3894          $LOG    TPL_LDS,TPL_PRSN_UXPK,,R5       ; Unexpected message
                    21    11    1225    3895          BRB     70$                             ; Log the event, bring adj down
                                1227    3896 20$:       ;
                                1227    3897          ;   Store partner's block size and listen timer parsed from message.
                                1227    3898          ;
 06 A7    00000018'EF     B0    1227    3899          MOVW    LEV_W_BLKSIZE,ADJ$W_BUFSIZ(R7)  ; Setup partner's buff size
    50    00000020'EF     3C    122F    3900          MOVZWL  LEV_W_HELLO,R0                  ; Get partner's hello timer
             50    03     C4    1236    3901          MULL    #TR4C_BCT3MULT,R0               ; Multiply by hello/listen factor
          08 A7    50     B0    1239    3902          MOVW    R0,ADJ$W_INT_LSN(R7)            ; Set listen interval
          0A A7    50     B0    123D    3903          MOVW    R0,ADJ$W_TIM_LSN(R7)            ; Start listen timer
             51    00     D0    1241    3904          MOVL    S^#LEV$C_NO_EVT,R1              ; No further events
             50    01     90    1244    3905          MOVB    #1,R0                           ; Allow state change
                          05    1247    3906          RSB
                                1248    3907
                                1248    3908 ;
                                1248    3909 ; Log event record & bring adjancency down
                                1248    3910 ;
             51    24     D0    1248    3911 70$:     MOVL    #LEV$C_LOG_ADE,R1               ; Signal "adjacency down event"
             50    01     D0    124B    3912          MOVL    #1,R0                           ; Make state change
                          05    124E    3913          RSB
```

```
                        124F  3915          .SBTTL  ACT_RCV_RT - Receive routing message
                        124F  3916 ;+
                        124F  3917 ; ACT_RCV_RT - React to received routing message
                        124F  3918 ; ACT_RCV_RTA - React to routing message received while in acceptance algorithm
                        124F  3919 ;
                        124F  3920 ; Inputs:        R11 = CRI CNR address
                        124F  3921 ;                R10 = CRI CNF address
                        124F  3922 ;                R7 = ADJ address
                        124F  3923 ;                R6 = LPD address
                        124F  3924 ;                R5 = WQE address
                        124F  3925 ;
                        124F  3926 ; Outputs:       R0 = True if state change requested
                        124F  3927 ;                R1 = Next event to be processed
                        124F  3928 ;
                        124F  3929 ;                R6 is the only register preserved.
                        124F  3930 ;-
                        124F  3931 ACT_RCV_RT:                                  ; React to rcv'd routing message
          19    10      124F  3932          BSBB    PROC_RT                   ; Do common processing
        09 50   E9      1251  3933          BLBC    R0,10$                    ; If LBC then something's wrong
          01D6  30      1254  3934          BSBW    REQUEST_UPDATE            ; Request running of update algorithm
    51    00    D0      1257  3935          MOVL    #LEV$C_NO_EVT,R1          ; No more events
    50    01    D0      125A  3936          MOVL    #1,R0                     ; Allow state change
                05      125D  3937 10$:     RSB                               ; Return state table control in R0/R1
                        125E  3938
                        125E  3939 ACT_RCV_RTA:                                 ; Receive routing message while running
                        125E  3940                                              ; the acceptance algorithm
          0A    10      125E  3941          BSBB    PROC_RT                   ; Do common processing
        06 50   E9      1260  3942          BLBC    R0,10$                    ; If LBC then something's wrong
                        1263  3943          ;
                        1263  3944          ;   Terminate the acceptance testing and generate a "circuit up"
                        1263  3945          ;   event.  This is necessary since we've just updated the matrix.
                        1263  3946          ;
          1A A6 94      1263  3947          CLRB    LPD$B_TSTCNT(R6)          ; Don't send any more test messages
    51    10    D0      1266  3948          MOVL    #LEV$C_LIN_UP,R1          ; Signal "circuit up"
                05      1269  3949 10$:     RSB                               ; Return state table control in R0/R1
                        126A  3950
                        126A  3951 PROC_RT:                                     ; Common Routing message processing
                        126A  3952          $DISPATCH LPD$B_ETY(R6),TYPE=B,<- ; If we are an endnode,
                        126A  3953                  <ADJ$C_PTY_PH4N,5$>,-     ; never process rtg messages
                        126A  3954                  <ADJ$C_PTY_PH3N,5$>>
                        1279  3955          ;
                        1279  3956          ;   Is the adjacency in the RUN state?  If not, ignore the routing
                        1279  3957          ;   message, since it might have preceeded the necessary Router
                        1279  3958          ;   Hello messages (for broadcast circuits ONLY).
                        1279  3959          ;
    0A 22 A6    0A      1279  3960          BBC     #LPD$V_BC,LPD$W_STS(R6),10$ ; If broadcast circuit,
       06 67    01      127E  3961          BBS     #ADJ$V_RUN,ADJ$B_STS(R7),10$ ; and if not in "run" state,
          51    00      1282  3962 5$:      MOVL    #LEV$C_NO_EVT,R1          ; Drop message - No more events
                50      1285  3963          CLRL    R0                        ; Indicate nothing happened
                05      1287  3964          RSB
                        1288  3965 10$:     ;
                        1288  3966          ;   Did the operator change the adjacent node's address?
                        1288  3967          ;
00000014'EF     04 A7   1288  3968          CMPW    ADJ$W_PNA(R7),LEV_W_PNA   ; Is the address the same as it was?
                6F 12   1290  3969          BNEQ    80$                       ; If NEQ then not the same
                        1292  3970          ;
                        1292  3971          ;   Determine if this is a Phase III or Phase IV routing message.
```

```
                                1292  3972          ;   Phase III routing messages are unsegmented, so we can copy
                                1292  3973          ;   the entire routing portion into the cost/hops buffer.  Phase IV
                                1292  3974          ;   routing messages are segmented, so we must run through each
                                1292  3975          ;   segment, copying the information into the right place.
                                1292  3976          ;
          50    20 A5   3C      1292  3977          MOVZWL  WQE$W_ADJ_INX(R5),R0    ; Get the ADJ index
    50  00000980'EF40   D0      1296  3978          MOVL    NET$AR_CH_VEC[R0],R0    ; Point to cost/hops buffer
                 57   13      129E  3979          BEQL    70$                     ; If none, then message error
          59    14 A5   3C      12A0  3980          MOVZWL  WQE$L_PM2(R5),R9        ; Get msg offset to routing info
             59    55   C0      12A4  3981          ADDL    R5,R9                   ; Convert to pointer
          58    16 A5   3C      12A7  3982          MOVZWL  WQE$L_PM2+2(R5),R8      ; Get number of bytes of rtginfo
    00  00000038'EF   91      12AB  3983          CMPB    PTYPE,#ADJ$C_PTY_PH3    ; Is it Phase III message?
                 0F   12      12B2  3984          BNEQ    50$                     ; If not, then Phase IV
                              12B4  3985          ;
                              12B4  3986          ;   If Phase III routing message, then copy the entire cost/hops
                              12B4  3987          ;   portion into the cost/hops buffer for this circuit.
                              12B4  3988          ;
             53    59   D0      12B4  3989          MOVL    R9,R3                   ; Set address of rtginfo
       51    58    FF   8F      12B7  3990          ASHL    #-1,R8,R1               ; Compute number of nodes
             52    01   D0      12BC  3991          MOVL    #1,R2                   ; Set starting node number
                   4F   10      12BF  3992          BSBB    UPDATE_MATRIX           ; Update the routing matrix
                   2D   11      12C1  3993          BRB     90$
                              12C3  3994          ;
                              12C3  3995          ;   If Phase IV routing message, the run through the segments,
                              12C3  3996          ;   copying each portion into the right place in the cost/hops buffer.
                              12C3  3997          ;
             58    04   C2      12C3  3998  50$:    SUBL    #4,R8                   ; Account for COUNT & STARTID
                   2F   15      12C6  3999          BLEQ    70$                     ; Branch if packet format error
             51    89   3C      12C8  4000          MOVZWL  (R9)+,R1                ; Get number of nodes in segment
             52    89   3C      12CB  4001          MOVZWL  (R9)+,R2                ; Get starting node number
             53    59   D0      12CE  4002          MOVL    R9,R3                   ; Set address of rtginfo
          7E 52    51   C1      12D1  4003          ADDL3   R1,R2,-(SP)             ; Compute ending+1 node number
00000400  8F       8E   D1      12D5  4004          CMPL    (SP)+,#NUM_NODES        ; Larger than our buffer?
                   19   1A      12DC  4005          BGTRU   70$                     ; If so, error in routing message
          7E 51    01   78      12DE  4006          ASHL    #1,R1,-(SP)             ; Compute number of bytes of rtginfo
             59    6E   C0      12E2  4007          ADDL    (SP),R9                 ; Skip past rtginfo
             58    8E   C2      12E5  4008          SUBL    (SP)+,R8                ; Account for cost/hops info
                   0D   19      12E8  4009          BLSS    70$                     ; Branch if packet format error
                   24   10      12EA  4010          BSBB    UPDATE_MATRIX           ; Update the routing matrix
                   58   D5      12EC  4011          TSTL    R8                      ; Anything more?
                   D3   14      12EE  4012          BGTR    50$                     ; If so, continue
             51    00   D0      12F0  4013  90$:    MOVL    #LEV$C_NO_EVT,R1        ; No more events
             50    01   D0      12F3  4014          MOVL    #1,R0                   ; Allow state change
                   05   12F6  4015          RSB
                              12F7  4016
                              12F7  4017
                              12F7  4018  ;
                              12F7  4019  ; Routing message format error
                              12F7  4020  ;
                              12F7  4021  70$:    $LOG    TPL_LDF,TPL_PRSN_RUCS,,R5       ; Log "checksum error"
                   08   11      12FF  4022          BRB     85$                     ; Log the event record
                              1301  4023  ;
                              1301  4024  ; Adjacent node address has changed - log event and bring line down
                              1301  4025  ;
                              1301  4026  80$:    $LOG    TPL_LDO,TPL_PRSN_ADJC,,R5       ; Assume address change
             51    24   D0      1309  4027  85$:    MOVL    #LEV$C_LOG_ADE,R1       ; Signal adjacency down event
             50    01   D0      130C  4028          MOVL    #1,R0                   ; Make state change
```

05 130F 4029 RSB

```
                                1310  4031          .SBTTL  UPDATE_MATRIX - Update the routing matrix
                                1310  4032 ;-
                                1310  4033 ; UPDATE_MATRIX - Update the routing matrix
                                1310  4034 ;
                                1310  4035 ; Inputs:
                                1310  4036 ;
                                1310  4037 ;       R0 = Base address of node column in routing matrix for the adjacency
                                1310  4038 ;       R1 = Number of nodes (non-zero)
                                1310  4039 ;       R2 = Starting node number
                                1310  4040 ;       R3 = Address of cost/hops routing information
                                1310  4041 ;       (R6 = LPD address, for journalling of routing changes)
                                1310  4042 ;       (R7 = ADJ address, for journalling of routing changes)
                                1310  4043 ;
                                1310  4044 ; Outputs:
                                1310  4045 ;
                                1310  4046 ;       None
                                1310  4047 ;
                                1310  4048 ;       The RTG_CHG vector is updated to reflect modifications
                                1310  4049 ;       to the "node column" of the routing matrix.
                                1310  4050 ;
                                1310  4051 ;       All registers are preserved.
                                1310  4052 ;-
                                1310  4053 UPDATE_MATRIX:
                      1F    BB  1310  4054          PUSHR   #^M<R0,R1,R2,R3,R4>     ; Save registers
                    54  6042  3E  1312  4055          MOVAW   (R0)[R2],R4             ; Address of 1st node in matrix
                50  00000000'EF D0  1316  4056          MOVL    NET$GL_PTR_VCB,R0       ; Get RCB address
                      18 A0   D5  131D  4057          TSTL    RCB$L_PTR_JNX(R0)       ; Is journalling enabled?
                      19      12  1320  4058          BNEQ    60$                     ; If so, use a slower loop
                                1322  4059          ;
                                1322  4060          ;   This loop is used when journalling is turned off, so that
                                1322  4061          ;   journalling doesn't slow down this loop when disabled.
                                1322  4062          ;
                   84   83     B1  1322  4063 10$:    CMPW    (R3)+,(R4)+             ; Same info as last message?
                        0D     13  1325  4064          BEQL    20$                     ; If so, no need to do anything
                FE A4    FE A3  B0  1327  4065          MOVW    -2(R3),-2(R4)           ; Store the changed cost/hops
                                132C  4066          SETBIT  R2,RTG_CHG              ; Update "node changes" vector
                        52      D6  1334  4067 20$:    INCL    R2                      ; Increment node number
                     E9 51      F5  1336  4068          SOBGTR  R1,10$                  ; Loop until all nodes done
                        35      11  1339  4069          BRB     90$
                                133B  4070          ;
                                133B  4071          ;   This loop is used when journalling is turned on.  The idea
                                133B  4072          ;   is to log all changes in routing information, so that using
                                133B  4073          ;   the journal, we can trace the routing activity of a node.
                                133B  4074          ;
                   84   83     B1  133B  4075 60$:    CMPW    (R3)+,(R4)+             ; Same info as last message?
                        2B     13  133E  4076          BEQL    70$                     ; If so, no need to do anything
                          ECBD' 30  1340  4077          BSBW    NET$JNX_CO              ; Initialize journalling co-routine
                      18 50     E9  1343  4078          BLBC    R0,65$                  ; Skip if not enabled for some reason
                81        04    90  1346  4079          MOVB    #^X04,(R1)+             ; Record type = routing change
                81    20 A6     90  1349  4080          MOVB    LPD$B_PTH_INX(R6),(R1)+ ; LPD index
                81    04 A7     B0  134D  4081          MOVW    ADJ$W_PNA(R7),(R1)+     ; Neighbor node issuing rtg msg
                81        52    B0  1351  4082          MOVW    R2,(R1)+                ; Node number
                81    FE A4     B0  1354  4083          MOVW    -2(R4),(R1)+            ; Old routing info
                81    FE A3     B0  1358  4084          MOVW    -2(R3),(R1)+            ; New routing info
                        9E      16  135C  4085          JSB     @(SP)+                  ; Log the journal record
                FE A4    FE A3  B0  135E  4086 65$:    MOVW    -2(R3),-2(R4)           ; Store the changed cost/hops
                                1363  4087          SETBIT  R2,RTG_CHG              ; Update "node changes" vector
```

```
                52   D6  136B  4088 70$:    INCL    R2                          ; Increment node number
             CB 51   F5  136D  4089          SOBGTR  R1,60$                      ; Loop until all nodes done
                1F   BA  1370  4090 90$:    POPR    #^M<R0,R1,R2,R3,R4>         ; Restore registers
                     05  1372  4091          RSB
```

```
                                1373  4093          .SBTTL  ACT_RCV_ART - Receive area routing message
                                1373  4094 ;+
                                1373  4095 ; ACT_RCV_ART - React to received area routing message
                                1373  4096 ; ACT_RCV_ARTA - React to area routing message received while in acceptance algorith
                                1373  4097 ;
                                1373  4098 ; Inputs:        R11 = CRI CNR address
                                1373  4099 ;                R10 = CRI CNF address
                                1373  4100 ;                R7 = ADJ address
                                1373  4101 ;                R6 = LPD address
                                1373  4102 ;                R5 = WQE address
                                1373  4103 ;
                                1373  4104 ; Outputs:       R0 = True if state change requested
                                1373  4105 ;                R1 = Next event to be processed
                                1373  4106 ;
                                1373  4107 ;                R6 is the only register preserved.
                                1373  4108 ;-
                                1373  4109 ACT_RCV_ART:                           ; React to rcv'd area routing message
          19   10               1373  4110          BSBB    PROC_ART             ; Do common processing
        09 50  E9               1375  4111          BLBC    R0,10$               ; If LBC then something's wrong
         00B2  30               1378  4112          BSBW    REQUEST_UPDATE       ; Request running of update algorithm
    51    00   D0               137B  4113          MOVL    #LEV$C_NO_EVT,R1     ; No more events
    50    01   D0               137E  4114          MOVL    #1,R0                ; Allow state change
               05               1381  4115 10$:     RSB                          ; Return state table control in R0/R1
                                1382  4116
                                1382  4117 ACT_RCV_ARTA:                         ; Receive routing message while running
                                1382  4118                                       ; the acceptance algorithm
          0A   10               1382  4119          BSBB    PROC_ART             ; Do common processing
        06 50  E9               1384  4120          BLBC    R0,10$               ; If LBC then something's wrong
                                1387  4121          ;
                                1387  4122          ;   Terminate the acceptance testing and generate a "circuit up"
                                1387  4123          ;   event.  This is necessary since we've just updated the matrix.
                                1387  4124          ;
        1A A6   94              1387  4125          CLRB    LPD$B_TSTCNT(R6)     ; Don't send any more test messages
    51    10   D0               138A  4126          MOVL    #LEV$C_LIN_UP,R1     ; Signal "circuit up"
               05               138D  4127 10$:     RSB                          ; Return state table control in R0/R1
                                138E  4128
                                138E  4129 PROC_ART:                           ; Common Routing message processing
                                138E  4130          $DISPATCH LPD$B_ETY(R6),TYPE=B,<- ; If we are an endnode,
                                138E  4131                  <ADJ$C_PTY_PH4N,5$>,-   ; never process rtg messages
                                138E  4132                  <ADJ$C_PTY_PH3N,5$>>
                                139D  4133          ;
                                139D  4134          ;   Is the adjacency in the RUN state?  If not, ignore the routing
                                139D  4135          ;   message, since it might have preceeded the necessary Router
                                139D  4136          ;   Hello messages (for broadcast circuits).
                                139D  4137          ;
    0A 22 A6    0A  E1          139D  4138          BBC     #LPD$V_BC,LPD$W_STS(R6),10$ ; If broadcast circuit,
       06 67    01  E0          13A2  4139          BBS     #ADJ$V_RUN,ADJ$B_STS(R7),10$ ; and if not in "run" state,
          51    00  D0          13A6  4140 5$:      MOVL    #LEV$C_NO_EVT,R1     ; No more events
                50  D4          13A9  4141          CLRL    R0                   ; Indicate nothing happened
                    05          13AB  4142          RSB
                                13AC  4143 10$:     ;
                                13AC  4144          ;   Did the operator change the adjacent node's address?
                                13AC  4145          ;
00000014'EF     04  A7  B1      13AC  4146          CMPW    ADJ$W_PNA(R7),LEV_W_PNA   ; Is the address the same as it was?
                    5F  12      13B4  4147          BNEQ    80$                  ; If NEQ then not the same
                                13B6  4148          ;
                                13B6  4149          ;   Copy the routing information into the buffer associated with
```

```
                                13B6  4150          ;   this routing adjacency.
                                13B6  4151          ;
          50    20 A5     3C    13B6  4152          MOVZWL  WQE$W_ADJ_INX(R5),R0     ; Get the ADJ index
50  00001A88'EF40     D0    13BA  4153          MOVL    NET$AR_AREA_CH[R0],R0   ; Point to cost/hops buffer
                47     13    13C2  4154          BEQL    70$                     ; If none, then message error
          59    14 A5     3C    13C4  4155          MOVZWL  WQE$L_PM2(R5),R9        ; Get msg offset to routing info
                59     55    13C8  4156          ADDL    R5,R9                   ; Convert to pointer
          58    16 A5     3C    13CB  4157          MOVZWL  WQE$L_PM2+2(R5),R8      ; Get number of bytes of rtginfo
                58     04    13CF  4158  50$:     SUBL    #4,R8                   ; Account for COUNT & STARTID
                37     15    13D2  4159          BLEQ    70$                     ; Branch if packet format error
                51     89    13D4  4160          MOVZWL  (R9)+,R1                ; Get number of nodes in segment
                52     89    13D7  4161          MOVZWL  (R9)+,R2                ; Get starting node number
          51    51     01    13DA  4162          ASHL    #1,R1,R1                ; Compute number of bytes of rtginfo
          52    52     01    13DE  4163          ASHL    #1,R2,R2                ; Compute offset to node's cost/hops
                58     51    13E2  4164          SUBL    R1,R8                   ; Account for cost/hops info
                24     19    13E5  4165          BLSS    70$                     ; Branch if packet format error
          7E    52     51    13E7  4166          ADDL3   R1,R2,-(SP)             ; Compute ending offset into buffer
   00000080 8F     8E    D1    13EB  4167          CMPL    (SP)+,#NUM_AREAS*2      ; Greater than size of buffer?
                17     1A    13F2  4168          BGTRU   70$                     ; If so, error in routing message
                33     BB    13F4  4169          PUSHR   #^M<R0,R1,R4,R5>        ; Save registers
        6042    69     51    13F6  4170          MOVC    R1,(R9),(R0)[R2]        ; Copy into cost/hops buffer
                33     BA    13FB  4171          POPR    #^M<R0,R1,R4,R5>        ; Restore registers
                59     51    13FD  4172          ADDL    R1,R9                   ; Skip past rtginfo
                58     D5    1400  4173          TSTL    R8                      ; Anything more?
                CB     14    1402  4174          BGTR    50$                     ; If so, continue
                51     00    D0    1404  4175  90$:     MOVL    #LEV$C_NO_EVT,R1        ; No more events
                50     01    D0    1407  4176          MOVL    #1,R0                   ; Allow state change
                             05    140A  4177          RSB
                                   140B  4178
                                   140B  4179
                                   140B  4180  ;
                                   140B  4181  ; Routing message format error
                                   140B  4182  ;
                                   140B  4183  70$:     $LOG    TPL_LDF,TPL_PRSN_RUCS,,R5       ; Log "checksum error"
                08    11           1413  4184          BRB     85$                     ; Log the event record
                                   1415  4185  ;
                                   1415  4186  ; Adjacent node address has changed - log event and bring line down
                                   1415  4187  ;
                                   1415  4188  80$:     $LOG    TPL_LDO,TPL_PRSN_ADJC,,R5       ; Assume address change
51              24    D0    141D  4189  85$:     MOVL    #LEV$C_LOG_ADE,R1       ; Signal adjacency down event
50              01    D0    1420  4190          MOVL    #1,R0                   ; Make state change
                            05    1423  4191          RSB
```

```
                                  1424          .SBTTL  REQUEST_UPDATE - Request update of routing database
                                  1424  ;+
                                  1424  ; REQUEST_UPDATE - Request running of the "update" algorithm
                                  1424  ;
                                  1424  ; This routine is called for all normal updates to the routing database.
                                  1424  ; It prevents the update algorithm from being run too often, and hogging
                                  1424  ; the machine, by using a supression timer.
                                  1424  ;
                                  1424  ; Inputs:
                                  1424  ;
                                  1424  ;       None
                                  1424  ;
                                  1424  ; Outputs:
                                  1424  ;
                                  1424  ;       None
                                  1424  ;
                                  1424  ;       R4-R6,R10-R11 are preserved.
                                  1424  ;-
                                  1424  ACT_REQ_UPDATE:                          ; Update database based on CRI change
                      07    10    1424          BSBB    REQUEST_UPDATE           ; Request update
                  51  00    D0    1426          MOVL    #LEV$C_NO_EVT,R1         ; No more events
                  50  01    D0    1429          MOVL    #1,R0                    ; Allow state change
                            05    142C          RSB
                                  142D
                                  142D  REQUEST_UPDATE:                          ; Request running of update algorithm
                                  142D          ;
                                  142D          ;   If the suppression timer is not already ticking exit and wait for
                                  142D          ;   it to fire.  Otherwise reset it and run the update algorithm.
                                  142D          ;
                                  142D          SETBIT  #RTG_V_UPD,RTGFLG        ; Remember request to update
        4B 00000040'EF    00    E2    1435          BBSS    #RTG_V_RUS,RTGFLG,20$    ; Exit if supression timer is ticking
              51   0201 8F    3C    143D          MOVZWL  #<<WQE$C_QUAL_RTG>@8>!-  ; Setup suppression timer i.d.
                                  1442                  NET$C_TID_RUS,R1         ;
                 52   89'AF    9E    1442          MOVAB   B^TIMER_RUS,R2           ; Setup action routine
                    0F80 8F    BB    1446          PUSHR   #^M<R7,R8,R9,R10,R11>    ; Save registers
           5B   00000000'EF    D0    144A          MOVL    NET$GL_CNR_LNI,R11       ; Set CNR address
           5A   00000000'EF    D0    1451          MOVL    NET$GL_PTR_LNI,R10       ; Set local CNF address
                                  1458          $GETFLD lni,l,rsi                ; Get routing suppression timer value
                  53   58    D0    1465          MOVL    R8,R3                    ; Move to another register
                    0F80 8F    BA    1468          POPR    #^M<R7,R8,R9,R10,R11>    ; Restore registers
                      03 50    E8    146C          BLBS    R0,10$                   ; If not set, provide default
                  53   01    D0    146F          MOVL    #1,R3
  53   00   53   00989680 8F    7A    1472  10$:    EMUL    #10*1000*1000,R3,#0,R3   ; Convert to standard VMS time
                       EB82'    30    147B          BSBW    WQE$RESET_TIM            ; Reset the routing suppression timer
                                  147E          ;
                                  147E          ;   Run the update algorithm on the data base.
                                  147E          ;
                                  147E          CLRBIT  #RTG_V_UPD,RTGFLG        ; Indicate update request satisfied
                      17    10    1486          BSBB    UPDATE                   ; Update the routing data base
                            05    1488  20$:    RSB
                                  1489
                                  1489  TIMER_RUS:                               ; Update suppression timer has fired
                       F8FC    30    1489          BSBW    KILL_WQE                 ; Deallocate the timer block
                                  148C          CLRBIT  #RTG_V_RUS,RTGFLG        ; Indicate timer no longer ticking
        02 00000040'EF    01    E1    1494          BBC     #RTG_V_UPD,RTGFLG,10$    ; If BS then update has been requested
                      8F    10    149C          BSBB    REQUEST_UPDATE           ; Perform the update & reset timer
                            05    149E  10$:    RSB
```

```
                                149F  4251          .SBTTL  UPDATE - Update database and neighbors
                                149F  4252 ;+
                                149F  4253 ; UPDATE - Update the routing data base
                                149F  4254 ;
                                149F  4255 ; Run the routing algorithm, update the routing data base, and
                                149F  4256 ; schedule routing message transmission to all routing nodes.
                                149F  4257 ;
                                149F  4258 ; INPUTS:        None
                                149F  4259 ;
                                149F  4260 ; OUTPUTS:       None
                                149F  4261 ;
                                149F  4262 ;                R4-R6,R10-R11 are preserved.
                                149F  4263 ;-
                                149F  4264 UPDATE:                                      ; Update the routing data base
                      0C70 8F   BB  149F  4265          PUSHR   #^M<R4,R5,R6,R10,R11>   ; Save registers
                54    00000000'EF D0  14A3  4266          MOVL    NET$GL_PTR_VCB,R4       ; Get RCB address
                                    14AA  4267          $DISPATCH RCB$B_ETY(R4),TYPE=B,<- ; Do the full decision if we are:
                                    14AA  4268                  <ADJ$C_PTY_AREA,5$>,-   ; A level 2 router
                                    14AA  4269                  <ADJ$C_PTY_PH4,5$>,-    ; A level 1 router
                                    14AA  4270                  <ADJ$C_PTY_PH3,5$>>     ; A Phase III router
                                    14BA  4271          ;
                                    14BA  4272          ;   If we are an endnode, then run a much shorter and simpler
                                    14BA  4273          ;   decision algorithm.
                                    14BA  4274          ;
                          05D9  30  14BA  4275          BSBW    ENDNODE_DECISION        ; Run endnode algorithm
                          00C6  31  14BD  4276          BRW     90$                     ; exit
                                    14C0  4277
                00000000'EF   16  14C0  4278 5$:      JSB     NET$GET_RTG3            ; Get routing info
                          05 50   E9  14C6  4279          BLBC    R0,9$                   ; Branch if error
                          6A A4   B5  14C9  4280          TSTW    RCB$W_MAX_RTG(R4)       ; Any routing adjacencies?
                             03   12  14CC  4281          BNEQ    10$                     ; Continue if so
                          00B5   31  14CE  4282 9$:       BRW     90$                     ; Nothing to do
                                    14D1  4283 10$:       ;
                                    14D1  4284          ;   If we are a level 2 router, then update the area database
                                    14D1  4285          ;
                54    00000000'EF   D0  14D1  4286          MOVL    NET$GL_PTR_VCB,R4       ; Get RCB address
                   03    008A C4   91  14D8  4287          CMPB    RCB$B_ETY(R4),#ADJ$C_PTY_AREA ; Are we level 2 router?
                                 53   12  14DD  4288          BNEQ    20$                     ; Skip if not
                5B    00000000'EF   D0  14DF  4289          MOVL    NET$GL_CNR_LNI,R11      ; Get LNI root
                5A    00000000'EF   D0  14E6  4290          MOVL    NET$GL_PTR_LNI,R10      ; Get LNI CNF
                                    14ED  4291          $GETFLD lni,l,amh               ; Fetch max hops field
                              35 50   E9  14FA  4292          BLBC    R0,20$                  ; Br on error
                0000002C'EF    58   9A  14FD  4293          MOVZBL  R8,MAX_HOPS             ; Store it
                                    1504  4294          $GETFLD lni,l,amc               ; Fetch max cost field
                              1E 50   E9  1511  4295          BLBC    R0,20$                  ; Br on error
                00000030'EF    58   3C  1514  4296          MOVZWL  R8,MAX_COST             ; Store it
                   58    008C C4   9A  151B  4297          MOVZBL  RCB$B_MAX_AREA(R4),R8   ; Get max area address
                                 58   D6  1520  4298          INCL    R8                      ; Get number of area addresses counting
                                    1522  4299                                          ; address #0
                   5A    20 B448   3E  1522  4300          MOVAW   @RCB$L_PTR_AOA(R4)[R8],R10 ; Point past last OA entry
             5B    00000900'EF48   3E  1527  4301          MOVAW   NET$AW_AREA_C_H[R8],R11 ; Point past last Cost/Hops entry
                          02CF   30  152F  4302          BSBW    AREA_DECISION           ; Update the area data base
                                    1532  4303 20$:       ;
                                    1532  4304          ;   Call the DECISION algorithm to update the level 1 forwarding database
                                    1532  4305          ;
                5B    00000000'EF   D0  1532  4306          MOVL    NET$GL_CNR_LNI,R11      ; Get LNI root
                5A    00000000'EF   D0  1539  4307          MOVL    NET$GL_PTR_LNI,R10      ; Get LNI CNF
```

```
                                  1540  4308          $GETFLD lni,l,mho                  ; Fetch max hops field
                  33 50     E9    154D  4309          BLBC    R0,50$                     ; Br on error
     0000002C'EF    58     9A    1550  4310          MOVZBL  R8,MAX_HOPS                ; Store it
                                  1557  4311          $GETFLD lni,l,mco                  ; Fetch max cost field
                  1C 50     E9    1564  4312          BLBC    R0,50$                     ; Br on error
     00000030'EF    58     3C    1567  4313          MOVZWL  R8,MAX_COST                ; Store it
           58    5A A4     3C    156E  4314          MOVZWL  RCB$W_MAX_ADDR(R4),R8      ; Get max node address
                 58        D6    1572  4315          INCL    R8                         ; Get number of node addresses counting
                                  1574  4316                                             ; address #0
        5A    1C B448      3E    1574  4317          MOVAW   @RCB$L_PTR_OA(R4)[R8],R10  ; Point past last OA entry
  5B   00000100'EF48       3E    1579  4318          MOVAW   NET$AW_MIN_C_H[R8],R11     ; Point past last Cost/Hops entry
                 08        10    1581  4319          BSBB    DECISION                   ; Update the data base
                                  1583  4320          ;
                                  1583  4321          ;   Send routing messages to our neighbors, if the database changed
                                  1583  4322          ;
                 03D9      30    1583  4323 50$:     BSBW    UPD_NEIGHBORS
              0C70 8F      BA    1586  4324 90$:     POPR    #^M<R4,R5,R6,R10,R11>      ; Restore registers
                           05    158A  4325          RSB
```

```
                                158B  4327          .SBTTL  DECISION - Update forwarding database
                                158B  4328 ;+
                                158B  4329 ; DECISION - Update the routing and forwarding databases.
                                158B  4330 ;
                                158B  4331 ; Inputs:
                                158B  4332 ;
                                158B  4333 ;       R11 = Address of last entry+1 of min. cost/hops buffer
                                158B  4334 ;       R10 = Address of last entry+1 of OA vector
                                158B  4335 ;       R8 = Ending address corresponding to last entry in vectors
                                158B  4336 ;       MAX_COST = Maximum cost value allowed for routing database
                                158B  4337 ;       MAX_HOPS = Maximum hops value allowed for routing database
                                158B  4338 ;       RTG_CHG = Vector which indicates which nodes must be processed.
                                158B  4339 ;
                                158B  4340 ; OUTPUTS:      None
                                158B  4341 ;
                                158B  4342 ;               All registers are destroyed.
                                158B  4343 ;-
                                158B  4344 DECISION:
                                158B  4345          ;
                                158B  4346          ;   See if we need to do anything at all.
                                158B  4347          ;
54   00000000'EF    D0          158B  4348          MOVL    NET$GL_PTR_VCB,R4       ; Get RCB address
     57    5A A4   3C          1592  4349          MOVZWL  RCB$W_MAX_ADDR(R4),R7   ; Get max address
           57  07   C0          1596  4350          ADDL    #7,R7                   ; Allow for roundoff
57   57    FD 8F   78          1599  4351          ASHL    #-3,R7,R7               ; Divide by bits/byte
00000080'EF    57   2D          159E  4352          CMPC5   R7,RTG_CHG,-            ; Is the entire RTG_CHG vector 0?
   6E   00    00               15A5  4353                  #0,#0,(SP)
                01   12        15A8  4354          BNEQ    5$                      ; If at least 1 bit set, do it
                     05        15AA  4355          RSB                             ; Otherwise, exit now
                               15AB  4356 5$:       ;
                               15AB  4357          ;   Record a journal record marking when we have started the
                               15AB  4358          ;   routing algorithms.
                               15AB  4359          ;
                  EA52'  30    15AB  4360          BSBW    NET$JNX_CO              ; Initialize journalling co-routine
                14 50    E9    15AE  4361          BLBC    R0,8$                   ; Skip if journalling not enabled
              81   02    90    15B1  4362          MOVB    #^X02,(R1)+             ; Record type = Starting algorithm
                   81    94    15B4  4363          CLRB    (R1)+                   ; spare byte
36   00  00000080'EF    57    2C    15B6  4364          MOVC5   R7,RTG_CHG,#0,#64-8-2,(R1) ; Journal the routing bitvector
                         61          15BF
              51   53    D0    15C0  4365          MOVL    R3,R1                   ; Update pointer past record
                   9E    16    15C3  4366          JSB     @(SP)+                  ; Log the journal record
                               15C5  4367 8$:       ;
                               15C5  4368          ;   Force the cost/hops for node #0 to always be re-evaluated
                               15C5  4369          ;   each time, because of the code at the bottom of the loop
                               15C5  4370          ;   which resets the "nearest level 2 router" based on the
                               15C5  4371          ;   adjacency for node #0.
                               15C5  4372          ;
                               15C5  4373          SETBIT  #0,RTG_CHG              ; Always re-evaluate node #0
                               15CD  4374          ;
                               15CD  4375          ;   Init registers, and start the loop
                               15CD  4376          ;
54   00000000'EF    D0          15CD  4377          MOVL    NET$GL_PTR_VCB,R4       ; Get RCB address
                0092   31       15D4  4378          BRW     100$                    ; Advance to the end of the loop
                               15D7  4379          ;
                               15D7  4380          ;   See if this node needs to be looked at.  If we haven't received
                               15D7  4381          ;   a routing message from any of our neighbors indicating that the
                               15D7  4382          ;   node cost/hops has changed since last time, then skip the node.
```

```
                                15D7  4383 ;
09 00000080'EF    58    E4      15D7  4384 10$:    BBSC    R8,RTG_CHG,15$          ; Lookup node if necessary
               5A   02    C2     15DF  4385          SUBL    #2,R10                  ; Skip past OA entry for node
               5B   02    C2     15E2  4386          SUBL    #2,R11                  ; Skip past min cost/hops for node
                  0081    31     15E5  4387          BRW     100$                    ; Else, skip the node entirely
                                15E8  4388 15$:     ;
                                15E8  4389          ;   Determine least cost path to this node
                                15E8  4390          ;
                     0115   30  15E8  4391          BSBW    FIND_PATH_TO_NODE       ; Find hops, costs, and adjacency
                                15EB  4392          ;
                                15EB  4393          ;   If the cost or hops to this node exceeds our maximums,
                                15EB  4394          ;   then declare the node unreachable.
                                15EB  4395          ;
     0000002C'EF    51    91     15EB  4396          CMPB    R1,MAX_HOPS             ; Is the node within range?
                      09    1A  15F2  4397          BGTRU   30$                     ; If GTRU then no
     00000030'EF    52    B1     15F4  4398          CMPW    R2,MAX_COST             ; Is the node within range?
                      02    1B  15FB  4399          BLEQU   40$                     ; If LEQU then yes
                      50    D4  15FD  4400 30$:     CLRL    R0                      ; Node is unreachable
                                15FF  4401 40$:     ;
                                15FF  4402          ;   Build the packed cost/hops field
                                15FF  4403          ;
                                15FF  4404          ASSUME  TR3V_RT_COST  EQ  0
                   0A   51    F0  15FF  4405          INSV    R1,#TR3V_RT_HOPS,-
                   52   05         1602  4406                  #TR3S_RT_HOPS,R2        ; Merge hops/cost
                                1604  4407          ASSUME  TR3S_RT_HOPS+TR3S_RT_COST EQ 15
              52    8000 8F    AA  1604  4408          BICW    #^X<8000>,R2            ; The high bit must be zero (Transport
                                1609  4409                                          ; architectural requirement)
                                1609  4410          ;
                                1609  4411          ;   If the node is now unreachable, then force the cost and hops
                                1609  4412          ;   to infinity, so that our neighbors realize the node is down now
                                1609  4413          ;   (they might have a higher maxcost, and wouldn't realize the
                                1609  4414          ;   node is unreachable until much later).
                                1609  4415          ;
                      50    D5  1609  4416          TSTL    R0                      ; Is the node reachable?
                      05    12  160B  4417          BNEQ    55$                     ; If not,
              52    7FFF 8F    3C  160D  4418          MOVZWL  #^X<7FFF>,R2            ; then make cost/hops infinite
                                1612  4419 55$:     ;
                                1612  4420          ;   Send routing msg only if  MINCOST or MINHOPS have changed.  If
                                1612  4421          ;   there has been a change in the node's reachability then record
                                1612  4422          ;   this fact so that it can be sent to the event logger.
                                1612  4423          ;
                                1612  4424          ASSUME  TR3S_RT_HOPS+TR3S_RT_COST EQ 15
          7B    8000 8F    AA  1612  4425          BICW    #^X<8000>,-(R11)        ; Ignore high bit
                6B   52    B1  1617  4426          CMPW    R2,(R11)                ; Was there a hops or cost change ?
                      4A    13  161A  4427          BEQL    90$                     ; If EQL then no
          7FFF 8F    52    B1  161C  4428          CMPW    R2,#^X<7FFF>            ; Is node currently unreachable?
                      07    1E  1621  4429          BGEQU   57$                     ; If GEQU yes, reachability change
          7FFF 8F    6B    B1  1623  4430          CMPW    (R11),#^X<7FFF>         ; Was node unreachable before?
                      08    1F  1628  4431          BLSSU   58$                     ; If LSSU no, no reachability change
                                162A  4432 57$:     SETBIT  R8,REACH_EVT            ; Indicate change in reachability status
                6B   52    B0  1632  4433 58$:     MOVW    R2,(R11)                ; Update the vector
       51   58     FB 8F    78  1635  4434          ASHL    #-LPD$C_SRM_SHFT,R8,R1  ; Compute SRM bit for this node
            52     5C A4    9A  163A  4435          MOVZBL  RCB$B_MAX_LPD(R4),R2    ; Get number of circuits
            53   28 B442    D0  163E  4436 60$:     MOVL    @RCB$L_PTR_LPD(R4)[R2],R3 ; Get LPD address
                      0A    18  1643  4437          BGEQ    65$                     ; Branch if slot not valid
          05 22 A3    04    E1  1645  4438          BBC     #LPD$V_RUN,LPD$W_STS(R3),65$ ; Branch if circuit not up
                                164A  4439                                          ; (skip ADJ$V_RTG check to save time)
```

```
                        164A          SETBIT  R1,LPD$G_SRM(R3)          ; Set SRM flag
        EC 52     F5    164F  65$:    SOBGTR  R2,60$                    ; Loop through all circuits
                        1652          ;
                        1652          ;   If the node is a direct partner, and is a Phase II adjacency,
                        1652          ;   then do not include the node in routing messages, to enforce
                        1652          ;   Phase II non-routing rules, but keep the OA vector pointing
                        1652          ;   to the output adjacency for the Phase II node.
                        1652          ;
           50     D5    1652          TSTL    R0                        ; Is node directly adjacent?
           10     13    1654          BEQL    70$                       ; If not, skip Phase II check
51   2C B440     D0    1656          MOVL    @RCB$L_PTR_ADJ(R4)[R0],R1 ; Get output ADJ address
  02   01 A1     91    165B          CMPB    ADJ$B_PTYPE(R1),#ADJ$C_PTY_PH2 ; Phase II direct adjacency?
           05     12    165F          BNEQ    70$                       ; Branch if not
6B   7FFF 8F     B0    1661          MOVW    #^X<7FFF>,(R11)           ; If so, don't include in routing msgs
                        1666  70$:    ;
                        1666          ;   Update the OA (output adjacency) vector.
                        1666          ;
      7A   50     B0    1666  90$:    MOVW    R0,-(R10)                 ; Update output adjacency to node
        02 58     F4    1669  100$:   SOBGEQ  R8,102$                   ; Loop for each node address
           03     11    166C          BRB     105$                      ; Loop finished - go on
         FF66     31    166E  102$:   BRW     10$                       ; Branch helper - continue looping
                        1671  105$:   ;
                        1671          ;   For entry #0 (nearest level 2 router), we must decide if we
                        1671          ;   are the nearest level 2 router.  If so, force node #0 to a cost
                        1671          ;   and hops of 0.  Else, leave the value as computed from neighbors.
                        1671          ;
                        1671          ;   If no other areas are reachable except our own, then it may mean
                        1671          ;   we are an isolated area router.  If this is the case, then allow
                        1671          ;   routing to continue by simply pretending we are a level 1 router,
                        1671          ;   and propagate the "nearest level 2 router" as determined by our
                        1671          ;   neighbors, and never use the AOA vector for forwarding (since we
                        1671          ;   may not really know the state of the level 2 network).
                        1671          ;
                        1671          CLRBIT  #RCB$V_LVL2,-             ; Assume we cannot do level 2 routing
                        1671                  RCB$B_STATUS(R4)
03   008A C4     91    1676          CMPB    RCB$B_ETY(R4),#ADJ$C_PTY_AREA ; Are we an area router?
           1F     12    167B          BNEQ    120$                      ; Skip if not
58   008C C4     9A    167D          MOVZBL  RCB$B_MAX_AREA(R4),R8     ; Get the maximum area number
     20 B448     B5    1682  110$:   TSTW    @RCB$L_PTR_AOA(R4)[R8]    ; Is any other area reachable?
           11     13    1686          BEQL    115$                      ; If not, continue searching
008B C4    58     91    1688          CMPB    R8,RCB$B_HOMEAREA(R4)     ; Our own area?
           0A     13    168D          BEQL    115$                      ; Skip our own area - it doesn't count
      6A   01     B0    168F          MOVW    #LPD$C_LOC_INX,(R10)      ; If so, mark "nearest" as local ADJ
           6B     B4    1692          CLRW    (R11)                     ; and send "nearest = 0 cost/hops"
                        1694          SETBIT  #RCB$V_LVL2,-             ; Tell Transport it can use the AOA
                        1694                  RCB$B_STATUS(R4)          ; vector - we are not isolated.
        E6 58     F5    1699  115$:   SOBGTR  R8,110$                   ; Loop thru all areas
                        169C  120$:   ;
                        169C          ;   Take the value for entry #0 (nearest level 2 router) and store
                        169C          ;   it in a special place in the RCB so that TRANSPORT can get at
                        169C          ;   it easily.  In addition, set OA(0) to point to ourselves, since
                        169C          ;   the internal convention is that node #0 refers to ourselves.
                        169C          ;
00AC C4    6A     B0    169C          MOVW    (R10),RCB$W_LVL2(R4)      ; Store path to nearest level 2 router
      6A   01     B0    16A1          MOVW    #LPD$C_LOC_INX,(R10)      ; Set "local" adjacency for node 0
                        16A4                                            ; (node 0 is the always the local node)
                        16A4          ;
```

```
                                16A4  4497          ;   Log an event for each node whose reachability status has changed.
                                16A4  4498          ;
          58    03FF  8F    3C  16A4  4499          MOVZWL  #NUM_NODES-1,R8         ; Setup max node address
40 00000000'EF    58    E5  16A9  4500 200$:       BBCC    R8,REACH_EVT,210$       ; If BS then reachability change
                  58    DD  16B1  4501          PUSHL   R8                      ; Save node address
    55    00000000'EF    9E  16B3  4502          MOVAB   NET$AB_EVT_WQE,R5       ; Point to common event WQE
    54    00000000'EF    D0  16BA  4503          MOVL    NET$GL_PTR_VCB,R4       ; Get RCB
    5B    00000000'EF    D0  16C1  4504          MOVL    NET$GL_CNR_NDI,R11      ; Get the NDI root block
                E935'    30  16C8  4505          BSBW    NET$LOCATE_NDI          ; Get node's CNF block
             12 A5       58  16CB  4506          MOVW    R8,WQE$W_REQIDT(R5)     ; Setup the node address
                008B C4  F0  16CF  4507          INSV    RCB$B_HOMEAREA(R4),-    ; using the current area
                      0A     16D3  4508                  #TR4$V_ADDR_AREA,-
             12 A5       06  16D4  4509                  #TR4$S_ADDR_AREA,WQE$W_REQIDT(R5)
                      00  90  16D7  4510          MOVB    #EVC$C_TPL_PSTS_RCH,-   ; Assume node is now reachable
                   1E A5      16D9  4511                  WQE$B_EVL_DT1(R5)       ;
                1C B448   B5  16DB  4512          TSTW    @RCB$L_PTR_OA(R4)[R8]   ; Is node now reachable?
                      04  12  16DF  4513          BNEQ    205$                    ; If NEQ then yes
                      01  90  16E1  4514          MOVB    #EVC$C_TPL_PSTS_URC,-   ; Signal "unreachable"
                   1E A5      16E3  4515                  WQE$B_EVL_DT1(R5)       ;
                010E 8F    B0  16E5  4516 205$:    MOVW    #EVC$C_TPL_RCH,-        ; Setup event logging code
                  1C A5        16E9  4517                  WQE$W_EVL_CODE(R5)      ;
                   E912'   30  16EB  4518          BSBW    NET$EVT_INTRAW          ; Log the event
                      58 8ED0  16EE  4519          POPL    R8                      ; Restore node address
                B5    58    F5  16F1  4520 210$:    SOBGTR  R8,200$                 ; Loop for each node
                                16F4  4521          ;
                                16F4  4522          ;   Record a journal record marking when we have finished the
                                16F4  4523          ;   routing algorithms.
                                16F4  4524          ;
                   E909'   30  16F4  4525          BSBW    NET$JNX_CO              ; Initialize journalling co-routine
                05 50      E9  16F7  4526          BLBC    R0,300$                 ; Skip if journalling not enabled
             81    03      90  16FA  4527          MOVB    #^X03,(R1)+             ; Record type = Ending algorithm
                   9E      16  16FD  4528          JSB     @(SP)+                  ; Log the journal record
                           05  16FF  4529 300$:    RSB                             ; Exit
```

```
                                1700  4531          .SBTTL  FIND_PATH_TO_NODE - Find least cost path to node
                                1700  4532 ;+
                                1700  4533 ; FIND_PATH_TO_NODE - Find least cost path to a node in our area
                                1700  4534 ;
                                1700  4535 ; Inputs:        R8 = Node address
                                1700  4536 ;                R4 = RCB address
                                1700  4537 ;
                                1700  4538 ; Outputs:       R1 = Number of hops to node
                                1700  4539 ;                R2 = Cost to node
                                1700  4540 ;                R0 = New ADJ index of path to node
                                1700  4541 ;
                                1700  4542 ;                R4 is preserved.
                                1700  4543 ;-
                                1700  4544 FIND_PATH_TO_NODE:
          0400 8F    BB         1700  4545          PUSHR   #^M<R10>                ; Save registers
               00    ED         1704  4546          CMPZV   #TR4$V_ADDR_DEST,-      ; Is this the local node?
    58    0E A4    0A           1706  4547                  #TR4$S_ADDR_DEST,RCB$W_ADDR(R4),R8
               0F    13         170A  4548          BEQL    3$                      ; Branch if so
          008D C4    B5         170C  4549          TSTW    RCB$W_ALIAS(R4)         ; Is there an alias?
               11    13         1710  4550          BEQL    5$                      ; If so,
               00    ED         1712  4551          CMPZV   #TR4$V_ADDR_DEST,-      ; Is this the alias node number?
    58    008D C4    0A           1714  4552                  #TR4$S_ADDR_DEST,RCB$W_ALIAS(R4),R8
               08    12         1719  4553          BNEQ    5$                      ; If not, proceed
          50   01    9A         171B  4554 3$:      MOVZBL  #LPD$C_LOC_INX,R0       ; Setup index for 'local' adjacency
               51    7C         171E  4555          CLRQ    R1                      ; Zero cost, hops
             00A6    31         1720  4556          BRW     100$                    ; and exit with success
                                1723  4557
          57   01    D0         1723  4558 5$:      MOVL    #1,R7                   ; Init adjacency index
               50    D4         1726  4559          CLRL    R0                      ; Assume unreachable
          51   01    CE         1728  4560          MNEGL   #1,R1                   ; Init min hops value to infinity
          52   01    CE         172B  4561          MNEGL   #1,R2                   ; Init min cost value to infinity
     59   2C B447    D0         172E  4562 7$:      MOVL    @RCB$L_PTR_ADJ(R4)[R7],R9 ; Get ADJ address
       0F 69   01    E1         1733  4563          BBC     #ADJ$V_RUN,ADJ$B_STS(R9),10$ ; Skip check if PNA not valid and
                                1737  4564                                          ; assume cost/hops applies to our area
               0A    EF         1737  4565          EXTZV   #TR4$V_ADDR_AREA,-      ; Get the area that cost/hops applies to
   59   04 A9    06           1739  4566                  #TR4$S_ADDR_AREA,ADJ$W_PNA(R9),R9
               07    13         173D  4567          BEQL    10$                     ; If area = 0, assume our area
       008B C4    59    91     173F  4568          CMPB    R9,RCB$B_HOMEAREA(R4)   ; Is it for our area?
               76    12         1744  4569          BNEQ    20$                     ; If not, skip this one
59   00000980'EF47    D0       1746  4570 10$:     MOVL    NET$AL_CH_VEC[R7],R9    ; Point to cost/hops buffer
               6C    13         174E  4571          BEQL    20$                     ; Skip if none for this circuit
          5A   57    D0         1750  4572          MOVL    R7,R10                  ; Remember ADJ index for this path
         59   6948    3E         1753  4573          MOVAW   (R9)[R8],R9             ; Point to entry for this node
                                1757  4574          ;
                                1757  4575          ;   Get the cost/hops for this node over this adjacency,
                                1757  4576          ;   and increase it by the hop for ourself.  Also compute
                                1757  4577          ;   the new cost for this path.
                                1757  4578          ;
     56   2C B447    D0         1757  4579          MOVL    @RCB$L_PTR_ADJ(R4)[R7],R6 ; Get address of ADJ block
       56   02 A6    9A         175C  4580          MOVZBL  ADJ$B_LPD_INX(R6),R6    ; Get LPD index
     56   28 B446    D0         1760  4581          MOVL    @RCB$L_PTR_LPD(R4)[R6],R6 ; Get address of LPD
       56   29 A6    9A         1765  4582          MOVZBL  LPD$B_COST(R6),R6       ; Get cost for this circuit
                                1769  4583
                                1769  4584          ASSUME  TR3V_RT_COST  EQ  0
                                1769  4585          ASSUME  TR3S_RT_COST  EQ 10
                                1769  4586
 53   69    FC00 8F    AB       1769  4587          BICW3   #^X<FC00>,(R9),R3       ; Get the cost value
```

```
                53   56   A0  176F  4588          ADDW    R6,R3                   ; Add in the circuit's cost
                     48   1D  1772  4589          BVS     20$                     ; If VS then cost is infinite
              53   02FF   8F  B1  1774  4590          CMPW    #^X<2FF>,R3             ; Has cost overflowed allowed limit?
                     41   1F  1779  4591          BLSSU   20$                     ; If LSSU then yes, it's not a minimum
                                177B  4592          ;
                                177B  4593          ;   For a broadcast circuit, the cost/hops buffer contains
                                177B  4594          ;   the state of all endnodes on that broadcast circuit.
                                177B  4595          ;   So, if the node is "reachable" over the broadcast circuit,
                                177B  4596          ;   then we have found the shortest path (by definition),
                                177B  4597          ;   and return success immediately.
                                177B  4598          ;
             5C A4      57   91  177B  4599          CMPB    R7,RCB$B_MAX_LPD(R4)    ; Is this a main LPD adjacency?
                           0E   1A  177F  4600          BGTRU   15$                     ; Branch if not
           56   28 B447   D0  1781  4601          MOVL    @RCB$L_PTR_LPD(R4)[R7],R6 ; Get LPD address
        04 22 A6     0A   E1  1786  4602          BBC     #LPD$V_BC,LPD$W_STS(R6),15$ ; Branch if not broadcast circuit
                     41   10  178B  4603          BSBB    FIND_ENDNODE_BEA        ; Put node's BEA index in R10
                     2D   13  178D  4604          BEQL    20$                     ; Branch if not found (& unlikely)
                                178F  4605 15$:      ;
                                178F  4606          ;   Check to see if this path is "less cost" than the previous
                                178F  4607          ;   minimum cost.  If so, remember this path as the best one.
                                178F  4608          ;   Use the node address of the adjacent node as a tiebreaker.
                                178F  4609          ;
                52   53   B1  178F  4610          CMPW    R3,R2                   ; Is cost value a new minumum ?
                     28   1A  1792  4611          BGTRU   20$                     ; If GEQU then no
                     11   1F  1794  4612          BLSSU   18$                     ; If LSSU then yes
           56   2C B447   D0  1796  4613          MOVL    @RCB$L_PTR_ADJ(R4)[R7],R6 ; Get address of new ADJ
           55   2C B440   D0  179B  4614          MOVL    @RCB$L_PTR_ADJ(R4)[R0],R5 ; Get address of old ADJ
        04 A5     04 A6   B1  17A0  4615          CMPW    ADJ$W_PNA(R6),ADJ$W_PNA(R5) ; Highest adj. node address
                     15   1B  17A5  4616          BLEQU   20$                     ; is the tiebreaker for equal costs
                     0A   EF  17A7  4617 18$:     EXTZV   #TR3V_RT_HOPS,-
           55   69   05       17A9  4618                  #TR3S_RT_HOPS,(R9),R5   ; Get the hops value
                     55   96  17AC  4619          INCB    R5                      ; Add in the hop to the adjacent node
                1F   55   91  17AE  4620          CMPB    R5,#^X<1F>              ; Has the max hops overflowed ?
                     09   1A  17B1  4621          BGTRU   20$                     ; If LSSU then yes, it's not a minimum
                52   53   B0  17B3  4622          MOVW    R3,R2                   ; Save new minimum cost to node
                51   55   90  17B6  4623          MOVB    R5,R1                   ; Save hops to node
                50   5A   D0  17B9  4624          MOVL    R10,R0                  ; Save output ADJ index for path
             53   6A A4      3C  17BC  4625 20$:     MOVZWL  RCB$W_MAX_RTG(R4),R3    ; Get number of routing adjacencies
          02 57     53   F3  17C0  4626          AOBLEQ  R3,R7,30$               ; Loop until done
                     03   11  17C4  4627          BRB     100$                    ; Exit with success
                 FF65     31  17C6  4628 30$:     BRW     7$                      ; Continue looping
                0400 8F   BA  17C9  4629 100$:    POPR    #^M<R10>                ; Restore registers
                          05  17CD  4630          RSB
                                17CE  4631
                                17CE  4632          ;
                                17CE  4633          ; Find BEA adjacency index to an endnode.
                                17CE  4634          ;
                                17CE  4635
                                17CE  4636 FIND_ENDNODE_BEA:
             5A      6A A4      3C  17CE  4637          MOVZWL  RCB$W_MAX_RTG(R4),R10   ; Get starting BEA index
                          20   11  17D2  4638          BRB     8$                      ; Start at NBRA+1
           56   2C B44A   D0  17D4  4639 5$:      MOVL    @RCB$L_PTR_ADJ(R4)[R10],R6 ; Get ADJ address
             17 66     00   E1  17D9  4640          BBC     #ADJ$V_INUSE,ADJ$B_STS(R6),8$ ; Skip if not active
                     0A   EF  17DD  4641          EXTZV   #TR4$V_ADDR_AREA,-      ; Get partner's area number
        55   04 A6     06       17DF  4642                  #TR4$S_ADDR_AREA,ADJ$W_PNA(R6),R5
                     07   13  17E3  4643          BEQL    6$                      ; If area = 0, assume our area
             008B C4   55   91  17E5  4644          CMPB    R5,RCB$B_HOMEAREA(R4)   ; Our area?
```

```
                08   12  17EA  4645          BNEQ    8$                              ; If not, skip this adjacency
                00   ED  17EC  4646 6$:      CMPZV   #TR4$V_ADDR_DEST,-              ; Does this BEA correspond to the node?
58   04 A6      0A         17EE  4647                  #TR4$S_ADDR_DEST,ADJ$W_PNA(R6),R8
                0A   13  17F2  4648          BEQL    15$                             ; If so, exit with R10 = BEA index
     56   68    A4   3C  17F4  4649 8$:      MOVZWL  RCB$W_MAX_ADJ(R4),R6            ; Get index of last BEA
     D8 5A      56   F3  17F8  4650          AOBLEQ  R6,R10,5$                       ; Loop thru all BEAs
                5A   D4  17FC  4651          CLRL    R10                             ; If BEA not found, skip this path
                         17FE  4652                                                  ; (& this should not happen)
                5A   D5  17FE  4653 15$:     TSTL    R10                             ; Return with PSL set
                05   1800  4654              RSB
```

```
                                1801  4656          .SBTTL  AREA_DECISION - Update area forwarding database
                                1801  4657 ;+
                                1801  4658 ; AREA_DECISION - Update the area routing and forwarding databases.
                                1801  4659 ;
                                1801  4660 ; Inputs:
                                1801  4661 ;
                                1801  4662 ;       R11 = Address of last entry+1 of AOA vector
                                1801  4663 ;       R10 = Address of last entry+1 of min. cost/hops buffer
                                1801  4664 ;       R8 = Ending address corresponding to last entry in vectors
                                1801  4665 ;       MAX_COST = Maximum cost value allowed for routing database
                                1801  4666 ;       MAX_HOPS = Maximum hops value allowed for routing database
                                1801  4667 ;
                                1801  4668 ; OUTPUTS:      None
                                1801  4669 ;
                                1801  4670 ;               All registers are destroyed.
                                1801  4671 ;-
                                1801  4672 AREA_DECISION:
        54  00000000'EF    D0  1801  4673          MOVL    NET$GL_PTR_VCB,R4       ; Get RCB address
                    006D    31  1808  4674          BRW     100$                    ; Advance to the end of the loop
                                180B  4675          ;
                                180B  4676          ;   Determine least cost path to this node
                                180B  4677          ;
                    00AF    30  180B  4678 10$:     BSBW    FIND_PATH_TO_AREA       ; Find hops, costs, and adjacency
                                180E  4679          ;
                                180E  4680          ;   If the cost or hops to this node exceeds our maximums,
                                180E  4681          ;   then declare the node unreachable.
                                180E  4682          ;
    0000002C'EF     51    91  180E  4683          CMPB    R1,MAX_HOPS             ; Is the node within range?
                      09    1A  1815  4684          BGTRU   30$                     ; If GTRU then no
    00000030'EF     52    B1  1817  4685          CMPW    R2,MAX_COST             ; Is the node within range?
                      02    1B  181E  4686          BLEQU   40$                     ; If LEQU then yes
                      50    D4  1820  4687 30$:     CLRL    R0                      ; Node is unreachable
                                1822  4688 40$:     ;
                                1822  4689          ;   Build the packed cost/hops field
                                1822  4690          ;
                                1822  4691          ASSUME  TR3V_RT_COST  EQ  0
            0A      51    F0  1822  4692          INSV    R1,#TR3V_RT_HOPS,-
                      52    05  1825  4693                  #TR3S_RT_HOPS,R2        ; Merge hops/cost
                                1827  4694          ASSUME  TR3S_RT_HOPS+TR3S_RT_COST EQ 15
        52      8000  8F    AA  1827  4695          BICW    #^X<8000>,R2            ; The high bit must be zero (Transport
                                182C  4696                                          ; architectural requirement)
                                182C  4697          ;
                                182C  4698          ;   If the area is now unreachable, then force the cost and hops
                                182C  4699          ;   to infinity, so that our neighbors realize the area is down now
                                182C  4700          ;   (they might have a higher maxcost, and wouldn't realize the
                                182C  4701          ;   area is unreachable until much later).
                                182C  4702          ;
                      50    D5  182C  4703          TSTL    R0                      ; Is the node reachable?
                      05    12  182E  4704          BNEQ    55$                     ; If not,
        52      7FFF  8F    3C  1830  4705          MOVZWL  #^X<7FFF>,R2            ; then make cost/hops infinite
                                1835  4706 55$:     ;
                                1835  4707          ;   Send routing msg only if  MINCOST or MINHOPS have changed.  If
                                1835  4708          ;   there has been a change in the node's reachability then record
                                1835  4709          ;   this fact so that it can be sent to the event logger.
                                1835  4710          ;
                                1835  4711          ASSUME  TR3S_RT_HOPS+TR3S_RT_COST EQ 15
        7B      8000  8F    AA  1835  4712          BICW    #^X<8000>,-(R11)        ; Ignore high bit
```

```
                6B    52    B1  183A  4713          CMPW    R2,(R11)                ; Was there a hops or cost change ?
                      36    13  183D  4714          BEQL    90$                     ; If EQL then no
              7FFF    8F    52  B1  183F  4715          CMPW    R2,#^X<7FFF>            ; Is node currently unreachable?
                      07    1E  1844  4716          BGEQU   57$                     ; If GEQU yes, reachability change
              7FFF    8F    6B  B1  1846  4717          CMPW    (R11),#^X<7FFF>         ; Was node unreachable before?
                      08    1F  184B  4718          BLSSU   58$                     ; If LSSU no, no reachability change
                            1A4D  4719  57$:        SETBIT  R8,REACH_EVT            ; Indicate change in reachability status
                6B    52    B0  1855  4720  58$:     MOVW    R2,(R11)                ; Update the vector
          51    58    FA    8F  78  1858  4721          ASHL    #-LPD$C_ASRM_SHFT,R8,R1 ; Compute SRM bit for this node
                52    5C    A4  9A  185D  4722          MOVZBL  RCB$B_MAX_LPD(R4),R2    ; Get number of circuits
             53    28  B442    D0  1861  4723  60$:     MOVL    @RCB$C_PTR_LPD(R4)[R2],R3 ; Get LPD address
                        0A    18  1866  4724          BGEQ    65$                     ; Branch if slot not valid
           05    22    A3    04  E1  1868  4725          BBC     #LPD$V_RUN,LPD$W_STS(R3),65$ ; Branch if circuit not up
                            186D  4726                                                ; (skip ADJ$V_RTG check to save time)
                            186D  4727          SETBIT  R1,LPD$G_ASRM(R3)       ; Set SRM flag
                  EC    52    F5  1872  4728  65$:     SOBGTR  R2,60$                  ; Loop through all circuits
                            1875  4729          ;
                            1875  4730          ;   Update the AOA (area output adjacency) vector.
                            1875  4731          ;
                7A    50    B0  1875  4732  90$:     MOVW    R0,-(R10)               ; Update output adjacency to node
                   90    58    F5  1878  4733  100$:    SOBGTR  R8,10$                  ; Loop for each node address
                7A    01    B0  187B  4734          MOVW    #LPD$C_LOC_INX,-(R10)   ; Use "local" adjacency for node 0
                            187E  4735                                                ; (node 0 is the always the local node)
                      7B    B4  187E  4736          CLRW    -(R11)                  ; Use 0 cost/hops for the local area
                            1880  4737
                            1880  4738          ;
                            1880  4739          ;   Log an event for each node whose reachability status has changed.
                            1880  4740          ;
                58    3F    3C  1880  4741          MOVZWL  #NUM_AREAS-1,R8         ; Setup max area address
       2E 00000000'EF       58  E5  1883  4742  200$:    BBCC    R8,REACH_EVT,210$       ; If BS then reachability change
                      58    DD  188B  4743          PUSHL   R8                      ; Save node address
          55    00000000'EF    9E  188D  4744          MOVAB   NET$AB_EVT_WQE,R5       ; Point to common event WQE
          54    00000000'EF    D0  1894  4745          MOVL    NET$GL_PTR_VCB,R4       ; Get RCB
                   12    A5    58  B0  189B  4746          MOVW    R8,WQE$W_REQIDT(R5)     ; Setup the area address
                      00    90  189F  4747          MOVB    #EVC$C_TPL_PSTS_RCH,-   ; Assume node is now reachable
                      1E    A5        18A1  4748          WQE$B_EVL_DT1(R5)       ;
                20  B448    B5  18A3  4749          TSTW    @RCB$C_PTR_AOA(R4)[R8]  ; Is node now reachable?
                      04    12  18A7  4750          BNEQ    205$                    ; If NEQ then yes
                      01    90  18A9  4751          MOVB    #EVC$C_TPL_PSTS_URC,-   ; Signal "unreachable"
                      1E    A5        18AB  4752          WQE$B_EVL_DT1(R5)       ;
                0111  8F    B0  18AD  4753  205$:    MOVW    #EVC$C_TPC_ACH,-        ; Setup "area reachability change"
                      1C    A5        18B1  4754          WQE$W_EVL_CODE(R5)      ;
                   E74A'    30  18B3  4755          BSBW    NET$EVT_INTRAW          ; Log the event
                      58  8ED0  18B6  4756          POPL    R8                      ; Restore node address
                  C7  58    F5  18B9  4757  210$:    SOBGTR  R8,200$                 ; Loop for each node
                            18BC  4758
                      05    18BC  4759          RSB                             ; Exit
```

```
                                18BD  4761          .SBTTL  FIND_PATH_TO_AREA - Find least cost path to area
                                18BD  4762 ;+
                                18BD  4763 ; FIND_PATH_TO_AREA - Find least cost path to area
                                18BD  4764 ;
                                18BD  4765 ; Inputs:       R8 = Area address
                                18BD  4766 ;               R4 = RCB address
                                18BD  4767 ;
                                18BD  4768 ; Outputs:      R1 = Number of hops to area
                                18BD  4769 ;               R2 = Cost to area
                                18BD  4770 ;               R0 = New ADJ index of path to area
                                18BD  4771 ;
                                18BD  4772 ;               R4 is preserved.
                                18BD  4773 ;-
                                18BD  4774 FIND_PATH_TO_AREA:
          0400 8F     BB        18BD  4775          PUSHR   #^M<R10>                ; Save registers
      008B C4   58    91        18C1  4776          CMPB    R8,RCB$B_HOMEAREA(R4)   ; Is this the local area ?
                08    12        18C6  4777          BNEQ    5$                      ; If so,
           50   01    9A        18C8  4778          MOVZBL  #LPD$C_LOC_INX,R0       ; Setup index for 'local' adjacency
                51    7C        18CB  4779          CLRQ    R1                      ; Zero cost, hops
                   008A   31    18CD  4780          BRW     100$                    ; and exit
                                18D0  4781
                57    01    D0  18D0  4782 5$:      MOVL    #1,R7                   ; Init adjacency index
                50          D4  18D3  4783          CLRL    R0                      ; Assume unreachable
                51    01    CE  18D5  4784          MNEGL   #1,R1                   ; Init min hops value to infinity
                52    01    CE  18D8  4785          MNEGL   #1,R2                   ; Init min cost value to infinity
     59   00001A88'EF47    D0  18DB  4786 7$:      MOVL    NET$AL_AREA_CH[R7],R9   ; Point to cost/hops buffer
                6D    13        18E3  4787          BEQL    20$                     ; Skip if none for this circuit
           5A   57    D0        18E5  4788          MOVL    R7,R10                  ; Remember ADJ index for this path
          59   6948    3E        18E8  4789          MOVAW   (R9)[R8],R9             ; Point to entry for this area
                                18EC  4790          ;
                                18EC  4791          ;   Get the cost/hops for this area over this adjacency,
                                18EC  4792          ;   and increase it by the hop for ourself.  Also compute
                                18EC  4793          ;   the new cost for this path.
                                18EC  4794          ;
      56   2C B447    D0        18EC  4795          MOVL    @RCB$L_PTR_ADJ(R4)[R7].R6 ; Get address of ADJ block
        56   02 A6    9A        18F1  4796          MOVZBL  ADJ$B_LPD_INX(R6),R6    ; Get LPD index
      56   28 B446    D0        18F5  4797          MOVL    @RCB$L_PTR_LPD(R4)[R6],R6 ; Get address of LPD
        56   29 A6    9A        18FA  4798          MOVZBL  LPD$B_COST(R6),R6       ; Get cost for this circuit
                                18FE  4799
                                18FE  4800          ASSUME  TR3V_RT_COST  EQ  0
                                18FE  4801          ASSUME  TR3S_RT_COST  EQ 10
                                18FE  4802
  53  69    FC00 8F     AB        18FE  4803          BICW3   #^X<FC00>,(R9),R3       ; Get the cost value
           53   56    A0        1904  4804          ADDW    R6,R3                   ; Add in the circuit's cost
                49    1D        1907  4805          BVS     20$                     ; If VS then cost is infinite
           53   02FF 8F     B1        1909  4806          CMPW    #^X<2FF>,R3             ; Has cost overflowed allowed limit?
                42    1F        190E  4807          BLSSU   20$                     ; If LSSU then yes, it's not a minimum
                                1910  4808          ;
                                1910  4809          ;   For a broadcast circuit, the cost/hops buffer contains
                                1910  4810          ;   the state of all endnodes on that broadcast circuit.
                                1910  4811          ;   So, if the area is "reachable" over the broadcast circuit,
                                1910  4812          ;   then we have found the shortest path (by definition),
                                1910  4813          ;   and return success immediately.
                                1910  4814          ;
        5C A4    57    91        1910  4815          CMPB    R7,RCB$B_MAX_LPD(R4)    ; Is this a main LPD adjacency?
                   0F    1A        1914  4816          BGTRU   15$                     ; Branch if not
      56   28 B447    D0        1916  4817          MOVL    @RCB$L_PTR_LPD(R4)[R7],R6 ; Get LPD address
```

```
05 22 A6    0A   E1  191B  4818          BBC     #LPD$V_BC,LPD$W_STS(R6),15$ ; Branch if not broadcast circuit
         FEAB    30  1920  4819          BSBW    FIND_ENDNODE_BEA            ; Put node's BEA index in R10
           2D    13  1923  4820          BEQL    20$                         ; Branch if not found (&& unlikely)
                     1925  4821 15$:     ;
                     1925  4822          ;   Check to see if this path is "less cost" than the previous
                     1925  4823          ;   minimum cost.  If so, remember this path as the best one.
                     1925  4824          ;   Use the node address of the adjacent node as a tiebreaker.
                     1925  4825          ;
      52    53   B1  1925  4826          CMPW    R3,R2                       ; Is cost value a new minumum ?
            28   1A  1928  4827          BGTRU   20$                         ; If GEQU then no
            11   1F  192A  4828          BLSSU   18$                         ; If LSSU then yes
56    2C B447    D0  192C  4829          MOVL    @RCB$L_PTR_ADJ(R4)[R7],R6   ; Get address of new ADJ
55    2C B440    D0  1931  4830          MOVL    @RCB$L_PTR_ADJ(R4)[R0],R5   ; Get address of old ADJ
04 A5    04 A6   B1  1936  4831          CMPW    ADJ$W_PNA(R6),ADJ$W_PNA(R5) ; Highest adj. node address
            15   1B  193B  4832          BLEQU   20$                         ; is the tiebreaker for equal costs
            0A   EF  193D  4833 18$:     EXTZV   #TR3V_RT_HOPS,-
55    69    05       193F  4834                  #TR3S_RT_HOPS,(R9),R5       ; Get the hops value
            55   96  1942  4835          INCB    R5                          ; Add in the hop to the adjacent node
      1F    55   91  1944  4836          CMPB    R5,#^X<1F>                  ; Has the max hops overflowed ?
            09   1A  1947  4837          BGTRU   20$                         ; If LSSU then yes, it's not a minimum
      52    53   B0  1949  4838          MOVW    R3,R2                       ; Save new minimum cost to area
      51    55   90  194C  4839          MOVB    R5,R1                       ; Save hops to area
      50    5A   D0  194F  4840          MOVL    R10,R0                      ; Save output ADJ index for path
   53    6A A4   3C  1952  4841 20$:     MOVZWL  RCB$W_MAX_RTG(R4),R3        ; Get number of routing adjacencies
   81 57    53   F3  1956  4842          AOBLEQ  R3,R7,7$                    ; Loop until done
      0400 8F    BA  195A  4843 100$:    POPR    #^M<R10>                    ; Restore registers
                 05  195E  4844          RSB
```

```
                                195F  4846          .SBTTL  UPD_NEIGHBORS - Schedule routing messages
                                195F  4847 ;+
                                195F  4848 ; UPD_NEIGHBORS - Schedule routing messages for neighboring nodes
                                195F  4849 ;
                                195F  4850 ; Schedule routing message transmission on all routing and broadcast LPD's.
                                195F  4851 ;
                                195F  4852 ; Inputs:        None
                                195F  4853 ;
                                195F  4854 ; OUTPUTS:       None
                                195F  4855 ;
                                195F  4856 ;                R0-R2 are destroyed.
                                195F  4857 ;-
                                195F  4858 UPD_NEIGHBORS:
            01D0 8F     BB  195F  4859          PUSHR   #^M<R4,R6,R7,R8>                ; Save registers
    54  00000000'EF     D0  1963  4860          MOVL    NET$GL_PTR_VCB,R4               ; Get RCB address
                            196A  4861          $DISPATCH RCB$B_ETY(R4),TYPE=B,<-       ; If we are an endnode,
                            196A  4862                  <ADJ$C_PTY_PH4N,200$>,-         ; never send rtg messages
                            196A  4863                  <ADJ$C_PTY_PH3N,200$>>
        00000000'EF     16  197A  4864          JSB     NET$GET_RTG3                    ; Get routing info
              06 50     E9  1980  4865          BLBC    R0,9$                           ; Branch if error
       58     5C A4     9A  1983  4866          MOVZBL  RCB$B_MAX_LPD(R4),R8            ; Get number of circuits
                 03     12  1987  4867          BNEQ    110$                            ; If nonzero, then go ahead
              0098      31  1989  4868 9$:      BRW     200$                            ; Skip entire thing
                            198C  4869          ;
                            198C  4870          ;   Schedule routing message transmission on all routing LPDs
                            198C  4871          ;
       56     28 B448     D0  198C  4872 110$:   MOVL    @RCB$L_PTR_LPD(R4)[R8],R6       ; Get address of LPD
                 13     18  1991  4873          BGEQ    113$                            ; Branch if slot not valid
    0E 22 A6     04     E1  1993  4874          BBC     #LPD$V_RUN,LPD$W_STS(R6),113$   ; Br if LPD's circuit inactive
      57     2C B448     D0  1998  4875          MOVL    @RCB$L_PTR_ADJ(R4)[R8],R7       ; Get address of ADJ block
    07 22 A6     0A     E0  199D  4876          BBS     #LPD$V_BC,LPD$W_STS(R6),115$    ; If broadcast circuit
       03 67     02     E0  19A2  4877          BBS     #ADJ$V_RTG,ADJ$B_STS(R7),115$   ; or if routing node, go ahead
              0073      31  19A6  4878 113$:    BRW     130$                            ; Else, skip this LPD entirely
                            19A9  4879          ;
                            19A9  4880          ;   Send area routing messages to adjacent area routers
                            19A9  4881          ;
                            19A9  4882          ASSUME  LPD$C_ASRM_SIZE EQ 1            ; && fix this
     03     008A C4     91  19A9  4883 115$:    CMPB    RCB$B_ETY(R4),#ADJ$C_PTY_AREA   ; Are we an area router?
                 2C     12  19AE  4884          BNEQ    117$                            ; If not, skip this
    06 22 A6     0A     E0  19B0  4885          BBS     #LPD$V_BC,LPD$W_STS(R6),116$    ; Skip check if broadcast circuit
       03     01 A7     91  19B5  4886          CMPB    ADJ$B_PTYPE(R7),#ADJ$C_PTY_AREA ; Is the neighbor an area rtr?
                 21     12  19B9  4887          BNEQ    117$                            ; If not, skip it
              5E A6     D5  19BB  4888 116$:    TSTL    LPD$G_ASRM(R6)                  ; Any area stuff to send?
                 1C     13  19BE  4889          BEQL    117$                            ; Branch if not
                 06     E2  19C0  4890          BBSS    #LPD$V_XMT_ART,-                ; Flag need to send area rtg msg
          1C 24 A6            19C2  4891                  LPD$B_XMTFLG(R6),118$           ; and defer if already in progress
    62 A6     5E A6     D0  19C5  4892          MOVL    LPD$G_ASRM(R6),LPD$G_XMT_ASRM(R6) ; Copy SRM flags for transmission
              5E A6     D4  19CA  4893          CLRL    LPD$G_ASRM(R6)                  ; and clear primary flags
              54 A6     96  19CD  4894          INCB    LPD$B_ASRM_POS(R6)              ; Make sure we don't start at the
                            19D0  4895                                                  ; same place in the bitmask each tim
                            19D0  4896                                                  ; (to prevent segment loss repetitio
                 01     90  19D0  4897          MOVB    #LPD$C_ASRM_SIZE,-              ; Set number of bits to check
                 55 A6      19D2  4898                  LPD$B_ASRM_LEFT(R6)
              50     00     D0  19D4  4899          MOVL    #LEV$C_NO_EVT,R0                ; Setup event
                   F381     30  19D7  4900          BSBW    SET_DLC_EVT                     ; Schedule LPD activity
                      05    11  19DA  4901          BRB     118$
                            19DC  4902 117$:    CLRBIT  #LPD$V_XMT_ART,LPD$B_XMTFLG(R6) ; Do not send level 2 msgs
```

```
                        19E1  4903          ;
                        19E1  4904          ;   Send level 1 routing messages to adjacent level 1 routers
                        19E1  4905          ;
 0F 22 A6    0A  E0     19E1  4906 118$:    BBS     #LPD$V_BC,LPD$W_STS(R6),119$     ; Skip check if broadcast circuit
             0A  EF     19E6  4907          EXTZV   #TR4$V_ADDR_AREA,-               ; Get area of partner node
51   04 A7     06             19E8  4908                  #TR4$S_ADDR_AREA,ADJ$W_PNA(R7),R1
               07  13    19EC  4909          BEQL    119$                             ; If area = 0, assume our area
  008B C4     51  91     19EE  4910          CMPB    R1,RCB$B_HOMEAREA(R4)            ; In our area?
               23  12     19F3  4911          BNEQ    120$                             ; If not, don't send level 1 msg
                        19F5  4912 119$:    ASSUME  LPD$C_SRM_SIZE EQ 32
          56 A6    D5     19F5  4913          TSTL    LPD$G_SRM(R6)                    ; Anything to send?
               1E  13     19F8  4914          BEQL    120$                             ; Branch if not
               04  E2     19FA  4915          BBSS    #LPD$V_XMT_RT,-                  ; Flag need to send routing msg
        1D 24 A6          19FC  4916                      LPD$B_XMTFLG(R6),130$        ; and defer if already in progress
5A A6     56 A6    D0     19FF  4917          MOVL    LPD$G_SRM(R6),LPD$G_XMT_SRM(R6) ; Copy SRM flags for transmission
          56 A6    D4     1A04  4918          CLRL    LPD$G_SRM(R6)                    ; and clear primary flags
          52 A6    96     1A07  4919          INCB    LPD$B_SRM_POS(R6)                ; Make sure we don't start at the
                        1A0A  4920                                                   ; same place in the bitmask each tim
                        1A0A  4921                                                   ; (to prevent segment loss repetitio
               20  90     1A0A  4922          MOVB    #LPD$C_SRM_SIZE,-                ; Set number of bits to check
          53 A6          1A0C  4923                  LPD$B_SRM_LEFT(R6)
        50    00  D0     1A0E  4924          MOVL    #LEV$C_NO_EVT,R0                 ; Setup event
            F347  30     1A11  4925          BSBW    SET_DLL_EVT                      ; Schedule LPD activity
               34  10     1A14  4926          BSBB    START_XRT                        ; Reset routing update timer
               04  11     1A16  4927          BRB     130$                             ; Continue
                        1A18  4928 120$:    CLRBIT  LPD$V_XMT_RT,LPD$B_XMTFLG(R6)    ; No need for routing msg
                        1A1C  4929
          02 58    F5     1A1C  4930 130$:    SOBGTR  R8,140$                          ; Loop for all LPDs
               03  11     1A1F  4931          BRB     200$                             ; Exit when loop completes
             FF68  31     1A21  4932 140$:    BRW     110$                             ; Continue looping
                        1A24  4933
        01D0 8F    BA     1A24  4934 200$:    POPR    #^M<R4,R6,R7,R8>                 ; Restore registers
               05  1A28  4935          RSB
```

```
                                1A29          .SBTTL  TIMER_XRT - Automatic routing update timer
                                1A29  ;+
                                1A29  ; TIMER_XRT - Routing update timer has expired
                                1A29  ;
                                1A29  ; Inputs:       R5 = WQE address
                                1A29  ;
                                1A29  ; Outputs:      None
                                1A29  ;
                                1A29  ;               The WQE is deallocated.
                                1A29  ;-
                                1A29  TIMER_XRT:                                  ; Entered when the routing timer fires
          58    12 A5     3C    1A29          MOVZWL  WQE$W_REQIDT(R5),R8         ; Get LPD index
                 F358     30    1A2D          BSBW    KILL_WQE                    ; Deallocate the timer block
                 126D     30    1A30          BSBW    NET$FIND_LPD                ; Locate LPD
                13 50     E9    1A33          BLBC    R0,90$                      ; If not found, just go away
                                1A36          ;
                                1A36          ;   Set all bits in the SRM bitmask for this circuit, so that
                                1A36          ;   when it comes time to update the neighbor, a complete update
                                1A36          ;   will be sent for all nodes.
                                1A36          ;
                                1A36          ASSUME  LPD$C_SRM_SIZE EQ 32
          56 A6     01     CE    1A36          MNEGL   #1,LPD$G_SRM(R6)            ; Force rtginfo for all nodes to be sent
                                1A3A          ASSUME  LPD$C_ASRM_SIZE EQ 1        ; && fix this
          5E A6     01     CE    1A3A          MNEGL   #1,LPD$G_ASRM(R6)           ; Force rtginfo for all areas to be sent
                                1A3E          ;
                                1A3E          ;   Send routing messages to all our neighbors which have the SRM
                                1A3E          ;   flags set, as we have done above.  If the decision algorithm
                                1A3E          ;   is scheduled to be run soon, then don't send the messages now,
                                1A3E          ;   since they may be out-of-date.  Instead, we rely on the fact
                                1A3E          ;   that routing messages are automatically sent to all our neighbors
                                1A3E          ;   after the algorithm is run.
                                1A3E          ;
  03 00000040'EF     00     E0    1A3E          BBS     #RTG_V_RUS,RTGFLG,90$       ; If decision pending, msgs will
                                1A46                                              ; be sent automatically after it runs
                 FF16     30    1A46          BSBW    UPD_NEIGHBORS               ; Else, explicitly send the messages
                          05    1A49  90$:    RSB
```

```
                                  1A4A          .SBTTL  Start automatic routing update timer
                                  1A4A  ;+
                                  1A4A  ; START_XRT - Start or reset the automatic routing update timer
                                  1A4A  ;
                                  1A4A  ; Inputs:
                                  1A4A  ;
                                  1A4A  ;       R6 = LPD address
                                  1A4A  ;
                                  1A4A  ; Outputs:
                                  1A4A  ;
                                  1A4A  ;       None
                                  1A4A  ;
                                  1A4A  ;       R0-R3 are destroyed.
                                  1A4A  ;-
                                  1A4A  START_XRT:                                ; Start routing update timer for LPD
                 0FF0 8F     BB   1A4A          PUSHR   #^M<R4,R5,R6,R7,R8,R9,R10,R11>  ; Save registers
          5B   00000000'EF   D0   1A4E          MOVL    NET$GL_CNR_LNI,R11      ; Get LNI root
          5A   00000000'EF   D0   1A55          MOVL    NET$GL_PTR_LNI,R10      ; Get LNI CNF
             09 22 A6   0A   E0   1A5C          BBS     #LPD$V_BC,LPD$W_STS(R6),10$ ; Branch if broadcast circuit
                                  1A61          $CNFFLD lni,l,rti,R9            ; Use non-broadcast routing timer
                        07   11   1A68          BRB     20$
                                  1A6A  10$:    $CNFFLD lni,l,brt,R9            ; Use broadcast routing timer
                     E58C'   30   1A71  20$:    BSBW    CNF$GET_FIELD           ; Get the routing timer value
                     1A 50   E9   1A74          BLBC    R0,90$                  ; No timer if parameter not set
          51   20 A6   10   78   1A77          ASHL    #16,LPD$W_PTH(R6),R1    ; Shift LPD index into REQIDT
             51   0202 8F   B0   1A7C          MOVW    #<<WQE$C_QUAL_RTG>@8>!- ; Set routing update timer i.d.
                                  1A81                  NET$C_TID_XRT,R1        ; into lower word
                52   A5 AF   9E   1A81          MOVAB   B^TIMER_XRT,R2          ; Setup action routine
  53   00   58   00989680 8F   7A   1A85          EMUL    #10*1000*1000,R8,#0,R3  ; Convert to standard VMS time
                     E56F'   30   1A8E          BSBW    WQE$RESET_TIM           ; Reset the routing update timer
                 0FF0 8F     BA   1A91  90$:    POPR    #^M<R4,R5,R6,R7,R8,R9,R10,R11>  ; Restore registers
                             05   1A95          RSB
```

```
                                1A96  5007          .SBTTL  ENDNODE_DECISION -  Endnode decision algorithm
                                1A96  5008 ;+
                                1A96  5009 ; ENDNODE_DECISION - Endnode decision algorithm
                                1A96  5010 ;
                                1A96  5011 ; This routine is called each time we want to run the decision algorithm
                                1A96  5012 ; and we are an endnode.  It simply ensures that the cost/hops to ourselves
                                1A96  5013 ; is zero, and chooses the least cost circuit as the "designated output
                                1A96  5014 ; adjacency".
                                1A96  5015 ;
                                1A96  5016 ; Inputs:
                                1A96  5017 ;
                                1A96  5018 ;       R4 = RCB address
                                1A96  5019 ;
                                1A96  5020 ; Outputs:
                                1A96  5021 ;
                                1A96  5022 ;       None
                                1A96  5023 ;
                                1A96  5024 ;       Registers R0-R3, R5-R8 are destroyed.
                                1A96  5025 ;-
                                1A96  5026 ENDNODE_DECISION:
                                1A96  5027          ;
                                1A96  5028          ;   Ensure that the cost/hops to ourselves is always 0.
                                1A96  5029          ;
                     00   EF   1A96  5030          EXTZV   #TR4$V_ADDR_DEST,-      ; Get the local node number
           50   0E A4   0A      1A98  5031                  #TR4$S_ADDR_DEST,RCB$W_ADDR(R4),R0
                     17   13    1A9C  5032          BEQL    10$                     ; If zero, then skip it
              00000100'EF40   B4 1A9E  5033          CLRW    NET$AW_MIN_C_H[R0]      ; Zero our own entry
                     00   EF   1AA5  5034          EXTZV   #TR4$V_ADDR_DEST,-      ; Get the alias node number
           50   008D C4   0A      1AA7  5035                  #TR4$S_ADDR_DEST,RCB$W_ALIAS(R4),R0
                     07   13    1AAC  5036          BEQL    10$                     ; If zero, then skip it
              00000100'EF40   B4 1AAE  5037          CLRW    NET$AW_MIN_C_H[R0]      ; Zero our own entry
                                1AB5  5038 10$:     ;
                                1AB5  5039          ;   Choose the least cost circuit as the "designated output
                                1AB5  5040          ;   circuit".
                                1AB5  5041          ;
                52   01   CE   1AB5  5042          MNEGL   #1,R2                   ; Init R2 to least cost so far
                     53   D4   1AB8  5043          CLRL    R3                      ; Init R3 to least cost DRT so far
             58   5C A4   9A   1ABA  5044          MOVZBL  RCB$B_MAX_LPD(R4),R8    ; Get # circuits
          56   28 B448   D0   1ABE  5045 20$:     MOVL    @RCB$C_PTR_LPD(R4)[R8],R6 ; Get address of LPD
                     18   18   1AC3  5046          BGEQ    30$                     ; Branch if slot not valid
                01   58   D1   1AC5  5047          CMPL    R8,#LPD$C_LOC_INX       ; Local LPD?
                     13   13   1AC8  5048          BEQL    30$                     ; Skip the local LPD
        0E 22 A6    04   E1   1ACA  5049          BBC     #LPD$V_RUN,LPD$W_STS(R6),30$ ; Br if inactive
             52   29 A6   91   1ACF  5050          CMPB    LPD$B_COST(R6),R2       ; Least cost circuit?
                     08   1E   1AD3  5051          BGEQU   30$                     ; If not, keep looking
             52   29 A6   9A   1AD5  5052          MOVZBL  LPD$B_COST(R6),R2       ; Save new least cost value
             53   2C A6   3C   1AD9  5053          MOVZWL  LPD$W_DRT(R6),R3        ; Save new least cost designated router
                DE   58   F5   1ADD  5054 30$:     SOBGTR  R8,20$                  ; Loop thru all circuits
         00AA C4   53   B0   1AE0  5055          MOVW    R3,RCB$W_DRT(R4)        ; Set DRT for all outgoing transmits
                                1AE5  5056                                          ; with an unspecified circuit
                     05   1AE5  5057          RSB
```

```
                     1AE6  5059          .SBTTL  ACT_ENT_MOP - Enter MOP state
                     1AE6  5060 ;+
                     1AE6  5061 ; ACT_ENT_MOP   - Circuit has entered MOP mode while in the "run" state
                     1AE6  5062 ;
                     1AE6  5063 ; This routine is called when a it is detected that the circuit has entered
                     1AE6  5064 ; the so called "maintenance mode" -- also known as the "service mode".
                     1AE6  5065 ; An NML process is created to service the circuit.
                     1AE6  5066 ;
                     1AE6  5067 ; INPUTS:        R11     CRI CNR ptr
                     1AE6  5068 ;                R10     CRI CNF ptr
                     1AE6  5069 ;                R6      LPD ptr
                     1AE6  5070 ;                R5      WQE address
                     1AE6  5071 ;                R4      RCB address
                     1AE6  5072 ;
                     1AE6  5073 ; OUTPUTS:       R5      Unchanged
                     1AE6  5074 ;                R1      Next event to be processed
                     1AE6  5075 ;                R0      Low bit set if state change is permitted,
                     1AE6  5076 ;                        Low bit clear to avoid state change
                     1AE6  5077 ;
                     1AE6  5078 ;                All other regs may be clobbered.
                     1AE6  5079 ;-
                     1AE6  5080 ACT_ENT_MOP:                             ; Put the circuit into a service substate
                     1AE6  5081          ;
                     1AE6  5082          ;   Notify the DLE module
                     1AE6  5083          ;
          E517'  30  1AE6  5084          BSBW    DLE$MOP_REQUEST          ; Handle "MOP mode" condition
                     1AE9  5085          ;
                     1AE9  5086          ;   Recycle the circuit
                     1AE9  5087          ;
     51    11    D0  1AE9  5088          MOVL    #LEV$C_LIN_DOWN,R1       ; Switch to line down event
     50    01    D0  1AEC  5089          MOVL    #1,R0                    ; Make state change
                 05  1AEF  5090          RSB
```

```
                        1AF0  5092          .SBTTL  ACT_DLL_UP - Datalink has initialized
                        1AF0  5093 ;+
                        1AF0  5094 ; ACT_DLL_UP  - The datalink has initialized
                        1AF0  5095 ;
                        1AF0  5096 ; This routine is called after datalink protocol initialization.  It chooses
                        1AF0  5097 ; one of three actions to take:
                        1AF0  5098 ;
                        1AF0  5099 ;
                        1AF0  5100 ;   1.  If the operator state is "off" then the circuit is undergoing restart
                        1AF0  5101 ;       in order to notify the partner node that it is shutting down.  In this
                        1AF0  5102 ;       case, the state change is pre-empted with the LEV$C_OPR_OFF event.
                        1AF0  5103 ;
                        1AF0  5104 ;   2.  Else, if the circuit substate is "service" then the routine pre-empts
                        1AF0  5105 ;       the state change and exits with the LEV$C_ENT_DLE event.
                        1AF0  5106 ;
                        1AF0  5107 ;   3.  Else, the  LPD is prepared to commence Transport layer initialization
                        1AF0  5108 ;       over the circuit.
                        1AF0  5109 ;
                        1AF0  5110 ;
                        1AF0  5111 ; INPUTS:        R11     CRI CNR ptr
                        1AF0  5112 ;                R10     CRI CNF ptr
                        1AF0  5113 ;                R9-R8   Scratch
                        1AF0  5114 ;                R7      ADJ address
                        1AF0  5115 ;                R6      LPD address
                        1AF0  5116 ;                R5      WQE address
                        1AF0  5117 ;                R4      RCB address
                        1AF0  5118 ;                R3-R0   Scratch
                        1AF0  5119 ;
                        1AF0  5120 ; OUTPUTS:       R5-R7   Preserved
                        1AF0  5121 ;                R1      Next event to be processed
                        1AF0  5122 ;                R0      Low bit set if state change is permitted,
                        1AF0  5123 ;                        Low bit clear to avoid state change
                        1AF0  5124 ;
                        1AF0  5125 ;                All other regs may be clobbered.
                        1AF0  5126 ;-
                        1AF0  5127 ACT_DLL_UP:                                  ; The datalink has initialized
                57      DD  1AF0  5128          PUSHL   R7                      ; Save ADJ address
                        1AF2  5129          ;
                        1AF2  5130          ;   If the operator state is "off" then we are going thru data-link
                        1AF2  5131          ;   re-init as a means to notify the opposite end of the circuit that
                        1AF2  5132          ;   the link is shutting down.
                        1AF2  5133          ;
                        1AF2  5134          $GETFLD  cri,l_sta                  ; Get the operator state
              0A 50     E9  1AFF  5135          BLBC     R0,10$                 ; If LBC then the same as "off"
                        1B02  5136          $DISPATCH R8,<-                     ; Case on operator state
                        1B02  5137              <NMA$C_STATE_OFF, 10$>,-
                        1B02  5138              <NMA$C_STATE_SER, 20$>,-
                        1B02  5139              <NMA$C_STATE_ON,  20$>,-
                        1B02  5140          >
            51  05       D0  1B0C  5141 10$:      MOVL    #LEV$C_OPR_OFF,R1      ; Generate "operator says off" event
                50      D4  1B0F  5142          CLRL     R0                     ; Prevent previously intended state
                        1B11  5143                                              ; transition
                65      11  1B11  5144          BRB      90$                    ; Take common exit
                        1B13  5145 20$:     ;
                        1B13  5146          ;   The operator is not shutting down the circuit.  Either init for
                        1B13  5147          ;   use by Transport, or give it to a direct-access server process.
                        1B13  5148          ;
```

```
                02   E1   1B13  5149          BBC     #LPD$V_DLE,-                    ; If BS then marked for direct access
          07 22 A6        1B15  5150                   LPD$W_STS(R6),30$               ; (state could be ON or SERVICE)
           51   1D   D0   1B18  5151          MOVL    #LEV$C_ENT_DLE,R1               ; Generate new event
                50   D4   1B1B  5152          CLRL    R0                              ; Prevent state change
                59   11   1B1D  5153          BRB     90$                             ; Take common exit
                          1B1F  5154          ;
                          1B1F  5155          ;  The datalink is undergoing a normal startup sequence.  Tell
                          1B1F  5156          ;  NETDRIVER about new LPD and schedule the Transport init messages.
                          1B1F  5157          ;
           50    05   D0   1B1F  5158  30$:   MOVL    #NETUPD$_DLL_ON,R0              ; Setup function code
                1207   30   1B22  5159          BSBW    TELL_NETDRIVER                  ; Tell NETDRIVER
              1B A6    96   1B25  5160          INCB    LPD$B_ASTCNT(R6)                ; Account for Rcv IRP queued to the
                          1B28  5161                                                  ; datalink by NETDRIVER on our behalf
                          1B28  5162          ;
                          1B28  5163          ;  If we have been forced into Phase II protocol, mark the
                          1B28  5164          ;  adjacency as Phase II now, so that the correct start msg
                          1B28  5165          ;  is sent.
                          1B28  5166          ;
            008A C4   90  1B28  5167          MOVB    RCB$B_ETY(R4),-                 ; Preset "our node type" for circuit
              1D A6       1B2C  5168                  LPD$B_ETY(R6)
                          1B2E  5169          $GETFLD cri,l,xpt                       ; Circuit transport protocol
             09 50   E9   1B3B  5170          BLBC    R0,50$                          ; Branch if not set
           57   6E   D0   1B3E  5171          MOVL    (SP),R7                         ; Restore ADJ address
                39   10   1B41  5172          BSBB    XPT_TO_PTY                      ; Translate XPT to node type
         1D A6   58   90   1B43  5173          MOVB    R8,LPD$B_ETY(R6)                ; Set "our node type" for circuit
                          1B47  5174  50$:    ;
                          1B47  5175          ;  If this is a broadcast circuit, then skip the start/verification
                          1B47  5176          ;  messages, and chain to another action routine, which will handle
                          1B47  5177          ;  broadcast circuit startup.
                          1B47  5178          ;
      07 22 A6    0A   E1   1B47  5179          BBC     #LPD$V_BC,LPD$W_STS(R6),31$ ; Branch if not broadcast circuit
           51   13   D0   1B4C  5180          MOVL    #LEV$C_BC_UP,R1                 ; Generate new event
                50   D4   1B4F  5181          CLRL    R0                              ; Prevent state change this time
                25   11   1B51  5182          BRB     90$
                          1B53  5183  31$:    ;
                          1B53  5184          ;  Schedule transmission of start/verification messages for a
                          1B53  5185          ;  non-broadcast circuit.
                          1B53  5186          ;
                          1B53  5187          $GETFLD cri,l,xpt                       ; Were we forced into a specific type?
               57   6E   D0   1B60  5188          MOVL    (SP),R7                         ; Restore ADJ address
                 04 50   E8   1B63  5189          BLBS    R0,32$                          ; If so, don't dally at all
                          1B66  5190          SETBIT  LPD$V_XMT_DALLY,-               ; Dally before sending 1st "start"
                          1B66  5191                  LPD$B_XMTFLG(R6)                ; so that we can adapt to remote node
                          1B6A  5192  32$:    BISB    #LPD$M_XMT_STR!-                ; Schedule "start" msg
                          1B6B  5193                   LPD$M_XMT_VRF!-                ; Schedule "verification" msg
                          1B6B  5194                   LPD$M_XMT_IDLE,-               ; Flag to detect when last msg was sent
          24 A6    0E       1B6B  5195                   LPD$B_XMTFLG(R6)               ;
                          1B6E  5196          ;
                          1B6E  5197          ;  Enter "on-starting" state
                          1B6E  5198          ;
                00   90   1B6E  5199          MOVB    #NMA$C_LINSS_STA,-              ; Enter "starting" substate
                  27 A6       1B70  5200                       LPD$B_SUB_STA(R6)      ;
               51   00   D0   1B72  5201          MOVL    #LEV$C_NO_EVT,R1                ; No more events
               50   01   90   1B75  5202          MOVB    #1,R0                           ; Allow state transition
                    57 8ED0   1B78  5203  90$:    POPL    R7                              ; Restore ADJ address
                          05   1B7B  5204          RSB
                          1B7C  5205
```

```
                                1B7C  5206 ;
                                1B7C  5207 ; Map TRANSPORT TYPE parameter to node type
                                1B7C  5208 ;
                                1B7C  5209 ; Inputs:
                                1B7C  5210 ;
                                1B7C  5211 ;       R8 = Transport type parameter value
                                1B7C  5212 ;
                                1B7C  5213 ; Outputs:
                                1B7C  5214 ;
                                1B7C  5215 ;       R8 = Corresponding node type (ADJ$C_PTY_xxx)
                                1B7C  5216 ;
                                1B7C  5217 XPT_TO_PTY:
                                1B7C  5218         $DISPATCH R8,<-
                                1B7C  5219                 <NMA$C_CIRXPT_PH2,40$>,- ; Force Phase II init
                                1B7C  5220                 <NMA$C_CIRXPT_PH3,42$>,- ; Force Phase III Routing init
                                1B7C  5221                 <NMA$C_CIRXPT_NR4,44$>> ; Force Phase IV endnode init
58   FF 8F   9A                1B86  5222         MOVZBL  #ADJ$C_PTY_UNR,R8       ; Unknown
        0D   11                1B8A  5223         BRB     48$
   58   02   90                1B8C  5224 40$:    MOVB    #ADJ$C_PTY_PH2,R8       ; Phase II
        08   11                1B8F  5225         BRB     48$
   58   00   90                1B91  5226 42$:    MOVB    #ADJ$C_PTY_PH3,R8       ; Routing III
        03   11                1B94  5227         BRB     48$
   58   05   90                1B96  5228 44$:    MOVB    #ADJ$C_PTY_PH4N,R8      ; Nonrouting IV
             05                1B99  5229 48$:    RSB
```

```
                        1B9A  5231          .SBTTL  DLE-related state changes
                        1B9A  5232 ;+
                        1B9A  5233 ; ACT_ENT_DLE   -  Tell server process that circuit is ready
                        1B9A  5234 ; ACT_EXI_SERV  -  Exit service state if needed
                        1B9A  5235 ; ACT_SYN_FAIL  -  The circuit failed to synchronize
                        1B9A  5236 ; ACT_INI_FAIL  -  I/O failure during Transport initialization.
                        1B9A  5237 ;
                        1B9A  5238 ; INPUTS:       R11     CRI CNR ptr
                        1B9A  5239 ;               R10     CRI CNF ptr
                        1B9A  5240 ;               R6      LPD ptr
                        1B9A  5241 ;               R5      WQE address
                        1B9A  5242 ;               R4      RCB address
                        1B9A  5243 ;
                        1B9A  5244 ; OUTPUTS:      R5      Unchanged
                        1B9A  5245 ;               R1      Next event to be processed
                        1B9A  5246 ;               R0      Low bit set if state change is permitted,
                        1B9A  5247 ;                       Low bit clear to avoid state change
                        1B9A  5248 ;
                        1B9A  5249 ;               All other regs may be clobbered.
                        1B9A  5250 ;-
                        1B9A  5251 ACT_ENT_DLE:                                 ; Tell server the circuit is ready
              03   E0   1B9A  5252          BBS     #LPD$V_ACCESS,-            ; If BS then circuit is accessed by
        1E 22 A6         1B9C  5253           LPD$W_STS(R6),40$                 ; server process
                        1B9F  5254          ;
                        1B9F  5255          ;   The circuit is up (or at least the driver thinks so) but there is
                        1B9F  5256          ;   no server process accessing the circuit.  Queue a receive to the
                        1B9F  5257          ;   circuit.  When the receive completes it will serve as a signal that
                        1B9F  5258          ;   the remote end of the circuit is requesting service.
                        1B9F  5259          ;
           51   03   9A  1B9F  5260          MOVZBL  #LEV$C_UNJAM,R1             ; Assume some I/O is pending
             1B A6   95  1BA2  5261          TSTB    LPD$B_ASTCNT(R6)            ; Any other I/O pending ?
                22   12  1BA5  5262          BNEQ    100$                        ; If NEQ yes, recycle the circuit
        51    80 8F   9A  1BA7  5263          MOVZBL  #128,R1                     ; Setup size of P1 buffer
              0F23    30  1BAB  5264          BSBW    NET$DLL_QIO_CO              ; Call co-routine to init WQE
    003C'C2     53   D0  1BAE  5265          MOVL    R3,WQE$C_LENGTH+P1(R2)      ; Point to buffer
0038'C2     80 8F   9A  1BB3  5266          MOVZBL  #128,WQE$C_LENGTH+P2(R2)    ; Setup buffer size
           50   00'  D0  1BB9  5267          MOVL    S^#IO$_READLBLK,R0          ; Setup I/O function
                05   1BBC  5268          RSB                                 ; Return to issue I/O, and exit
                        1BBD  5269          ;
                        1BBD  5270          ;   The circuit is already being accessed by a server process.  Tell
                        1BBD  5271          ;   the circuit access module that the circuit is up and then tell
                        1BBD  5272          ;   NETDRIVER to start its receiver.
                        1BBD  5273          ;
           50   00'  D0  1BBD  5274 40$:     MOVL    S^#SS$_NORMAL,R0            ; Indicate success
              E43D' 30  1BC0  5275          BSBW    DLE$LPD_STATUS              ; Tell DLE module of circuit transition
           51   00   D0  1BC3  5276          MOVL    #LEV$C_NO_EVT,R1            ; No more events
           50   01   D0  1BC6  5277          MOVL    #1,R0                       ; Allow state change
                05   1BC9  5278 100$:    RSB
                        1BCA  5279
                        1BCA  5280 ACT_EXI_SERV:                                ; Exit service state if needed
              02   E1  1BCA  5281          BBC     #LPD$V_DLE,-                ; If not marked for direct-access then
        14 22 A6         1BCC  5282           LPD$W_STS(R6),10$                 ; nothing to do
              03   E0  1BCF  5283          BBS     #LPD$V_ACCESS,-             ; If currently being accessed then
        0F 22 A6         1BD1  5284           LPD$W_STS(R6),10$                 ; allow operation to complete
                        1BD4  5285
                        1BD4  5286          BUG_CHECK NETNOSTATE                ; && What are we doing here??
                        1BD8  5287          CLRBIT  LPD$V_DLE,LPD$W_STS(R6)     ; Else clear direct-access flag
```

```
                51    04    D0  1BDC  5288          MOVL    #LEV$C_REQ_SHUT,R1      ; Chain to "request shutdown" event
                50    00'   D0  1BDF  5289          MOVL    S^#SS$_NORMAL,R0        ; Allow state change
                            05  1BE2  5290          RSB
                51    00    D0  1BE3  5291  10$:     MOVL    #LEV$C_NO_EVT,R1        ; No further events
                50    00'   D0  1BE6  5292          MOVL    S^#SS$_NORMAL,R0        ; Allow state change
                            05  1BE9  5293          RSB
                                1BEA  5294
                                1BEA  5295  ACT_SYN_FAIL:                           ; The circuit failed to synchronize
                      03    E1  1BEA  5296          BBC     #LPD$V_ACCESS,-         ; If BC then circuit is not being "accessed"
             08 22 A6           1BEC  5297           LPD$W_STS(R6),10$              ; for direct-link sevice
          50  0000'8F    3C    1BEF  5298          MOVZWL  #SS$_DEVINACT,R0        ; "circuit no longer active"
              E409'       30  1BF4  5299          BSBW    DLE$LPD_STATUS          ; Tell DLE module of circuit transition
                51    04    D0  1BF7  5300  10$:     MOVL    #LEV$C_REQ_SHUT,R1      ; Chain to "request shutdown" event
                50    00'   D0  1BFA  5301          MOVL    S^#SS$_NORMAL,R0        ; Allow state change
                            05  1BFD  5302          RSB
                                1BFE  5303
                                1BFE  5304  ACT_INI_FAIL:                           ; I/O failure during transport init
                                1BFE  5305          BUMP    B,LPD$B_CNT_IFL(R6)     ; Increment circuit init failure count
                51    11    D0  1C07  5306          MOVL    #LEV$C_LIN_DOWN,R1      ; Signal "circuit down" event
                      50    D4  1C0A  5307          CLRL    R0                      ; Do not change state for this event
                            05  1C0C  5308          RSB
                                1C0D  5309
                                1C0D  5310  ACT_X25_RESET:                         ; X.25 "reset"
                      51    D4  1C0D  5311          CLRL    R1                      ; No QIO buffer needed
                   0EBF     30  1C0F  5312          BSBW    NET$DLL_QIO_CO          ; Allocate and init WQE (co-routine)
       0030'C2     03    D0  1C12  5313          MOVL    #PSI$C_RESET,WQE$C_LENGTH+P4(R2) ; Set P4 to "reset confirmation"
         10 A2     00    90  1C17  5314          MOVB    #LEV$C_NO_EVT,WQE$B_EVT(R2) ; Do nothing when I/O completes
         14 A2     00    90  1C1B  5315          MOVB    #LEV$C_NO_EVT,WQE$L_PM2(R2) ; Do nothing if I/O fails
          50  0000'8F    3C  1C1F  5316          MOVZWL  #IO$_NETCONTROL,R0      ; Set I/O function code
                            05  1C24  5317          RSB                         ; Issue I/O and exit
```

```
                     1C25  5319          .SBTTL  ACT_RUN_DOWN, ACT_SET_OPER
                     1C25  5320 ;+
                     1C25  5321 ; ACT_RUN_DOWN - Run down a circuit
                     1C25  5322 ; ACT_SET_OPER - Restart a stalled circuit
                     1C25  5323 ;
                     1C25  5324 ; INPUTS:        R11     CRI CNR ptr
                     1C25  5325 ;                R10     CRI CNF ptr
                     1C25  5326 ;                R6      LPD ptr
                     1C25  5327 ;                R5      WQE address
                     1C25  5328 ;                R4      RCB address
                     1C25  5329 ;
                     1C25  5330 ; OUTPUTS:       R5      Unchanged
                     1C25  5331 ;                R1      Next "event longword" to be processed
                     1C25  5332 ;                R0      Low bit set if state change is permitted,
                     1C25  5333 ;                        Low bit clear to avoid state change
                     1C25  5334 ;
                     1C25  5335 ;                All other registers may be clobbered
                     1C25  5336 ;-
                     1C25  5337 ACT_FAILED:
          0B   90    1C25  5338          MOVB    #NMA$C_LINSS_FAI,-      ; Mark outgoing call "failed"
      27 A6          1C27  5339          LPD$B_SUB_STA(R6)       ; (requiring operator intervention)
    51   00   D0     1C29  5340          MOVL    #LEV$C_NO_EVT,R1        ; No more events
    50   01   D0     1C2C  5341          MOVL    #1,R0                   ; Allow state change (to S state)
                05   1C2F  5342          RSB
                     1C30  5343
                     1C30  5344 ACT_RUN_DOWN:                           ; Cancel all timers, etc.
  51   20 A6   9A     1C30  5345          MOVZBL  LPD$B_PTH_INX(R6),R1    ; Get LPD index
51   51   10   78     1C34  5346          ASHL    #16,R1,R1               ; Shift into upper word
51   0100 8F   B0     1C38  5347          MOVW    #WQE$C_QUAL_DLL@8,R1    ; Setup QUAL, zero EVT for cancel all
       E3C0'   30     1C3B  5348          BSBW    WQE$CANCEL_TIM          ; Cancel all timers for the LPD cell
                     1C40  5349          CLRBIT  LPD$V_STRTIM,-          ; Start suppression timer is no longer
                     1C40  5350                  LPD$W_STS(R6)           ; ticking
                     1C44  5351          $GETFLD cri,l_sta               ; Get "operator" state
      05 50    E9     1C51  5352          BLBC    R0,10$                  ; If LBC then assume OFF
   58   01    91     1C54  5353          CMPB    #NMA$C_STATE_OFF,R8     ; Is it OFF ?
        0F    12     1C57  5354          BNEQ    50$                     ; If NEQ no
                     1C59  5355 10$:     ;
                     1C59  5356          ;   The operator is turning the line off.
                     1C59  5357          ;
1E 22 A6    03    E1     1C59  5358          BBC     #LPD$V_ACCESS,LPD$W_STS(R6),100$ ; If server process active,
 50   0000'8F    3C     1C5E  5359          MOVZWL  #SS$_DEVINACT,R0        ; "circuit no longer active"
       E39A'    30     1C63  5360          BSBW    DLE$LPD_STATUS          ; Tell DLE module of circuit transition
        14      11     1C66  5361          BRB     100$                    ; Continue
                     1C68  5362          ;
                     1C68  5363          ;   If the circuit substate has been marked "failed" (as a
                     1C68  5364          ;   result of "maximum recalls" exceeded), then do not allow
                     1C68  5365          ;   further circuit startup attempts until the operator explicitly
                     1C68  5366          ;   turns the circuit on (which clears substate).
                     1C68  5367          ;
      27 A6    91     1C68  5368 50$:     CMPB    LPD$B_SUB_STA(R6),-     ; "failed" circuit?
         0B           1C6B  5369                  #NMA$C_LINSS_FAI
         0E    13     1C6C  5370          BEQL    100$                    ; If so, stay in this state
                     1C6E  5371                                          ; until operator intervention
                     1C6E  5372          ;
                     1C6E  5373          ;   The circuit is entering a stalled state waiting for a server
                     1C6E  5374          ;   process to start some activity.  Set a timer so that we don't wait
                     1C6E  5375          ;   for ever.
```

```
                                  1C6E  5376          ;
53   00000000 23C34600 8F    7D   1C6E  5377          MOVQ    #60*<10*1000*1000>,R3   ; Wait 60 seconds
                     0FA4    30   1C79  5378          BSBW    SET_IOTIM               ; Start the timer
                  51   01    D0   1C7C  5379 100$:    MOVL    #LEV$C_EXIT,R1          ; No further events
                  50   01    D0   1C7F  5380          MOVL    #1,R0                   ; Allow state transition
                             05   1C82  5381          RSB
                                  1C83  5382
                                  1C83  5383
                                  1C83  5384 ACT_SET_OPER:                           ; Restart a stalled line
            08 22 A6   03    E1   1C83  5385          BBC     #LPD$V_ACCESS,LPD$W_STS(R6),100$ ; If server process active,
             50   0000'8F    3C   1C88  5386          MOVZWL  #SS$_DEVINACT,R0        ; "circuit no longer active"
                     E370'   30   1C8D  5387          BSBW    DLE$LPD_STATUS          ; Tell DLE module of circuit transition
                                  1C90  5388 100$:    $GETFLD cri,l,sta               ; Get "operator" state
             51   00FC'C8    9A   1C9D  5389          MOVZBL  OPR_EVT_MAP(R8),R1      ; Get corresponding event
                  50   01    D0   1CA2  5390          MOVL    #1,R0                   ; Allow state change
                             05   1CA5  5391          RSB                             ; Process new event
```

```
                                1CA6  5393          .SBTTL  ACT_TST_DL - Circuit acceptance algorithm
                                1CA6  5394 ;+
                                1CA6  5395 ; ACT_TST_DL  - Run circuit acceptance algorithm
                                1CA6  5396 ;
                                1CA6  5397 ; INPUTS:        R11     CRI CNR ptr
                                1CA6  5398 ;                R10     CRI CNF ptr
                                1CA6  5399 ;                R7      ADJ address
                                1CA6  5400 ;                R6      LPD address
                                1CA6  5401 ;                R5      WQE address
                                1CA6  5402 ;                R4      RCB address
                                1CA6  5403 ;
                                1CA6  5404 ; OUTPUTS:       R5-R7   Preserved
                                1CA6  5405 ;                R1      Next event to be processed
                                1CA6  5406 ;                R0      Low bit set if state change is permitted,
                                1CA6  5407 ;                        Low bit clear to avoid state change
                                1CA6  5408 ;
                                1CA6  5409 ;                All other regs may be clobbered.
                                1CA6  5410 ;-
                                1CA6  5411 ACT_TST_DL:                                  ; Run circuit acceptance algorithm
    0E0A    30                  1CA6  5412          BSBW    CHK_IO                      ; Okay to xmit?
51    01    D0                  1CA9  5413          MOVL    #LEV$C_EXIT,R1              ; Assume we cannot xmit
  08  50    E9                  1CAC  5414          BLBC    R0,10$                      ; If LBC then no
  1A  A6    95                  1CAF  5415          TSTB    LPD$B_TSTCNT(R6)            ; Any test messages to xmit?
      07    1A                  1CB2  5416          BGTRU   20$                         ; If so, send one
51    10    D0                  1CB4  5417          MOVL    #LEV$C_LIN_UP,R1            ; Signal circuit up event
50    01    90                  1CB7  5418 10$:     MOVB    #1,R0                       ; Always allow state change
            05                  1CBA  5419          RSB
                                1CBB  5420
                                1CBB  5421          ;
                                1CBB  5422          ;    Allocate and setup the buffer
                                1CBB  5423          ;
          51    83  8F    9A    1CBB  5424 20$:     MOVZBL  #1+2+1+TR3C_TST_MAX,R1      ; Setup max test size
          06  A7    51    B1    1CBF  5425          CMPW    R1,ADJ$W_BUFSIZ(R7)         ; Too big?
                      04    1B  1CC3  5426          BLEQU   30$                         ; If LEQU then no
          51    06  A7    3C    1CC5  5427          MOVZWL  ADJ$W_BUFSIZ(R7),R1         ; Use partner's rcv buf size
       58   51        04    C3  1CC9  5428 30$:     SUBL3   #1+2+T,R1,R8                ; Save size of test data field
                0E01    30    1CCD  5429          BSBW    NET$DLL_QIO_CO              ; Call co-routine to allocate buffer
     003C'C2    53    D0    1CD0  5430          MOVL    R3,WQE$C_LENGTH+P1(R2)      ; Point to I/O buffer
     0038'C2    53    CE    1CD5  5431          MNEGL   R3,WQE$C_LENGTH+P2(R2)      ; Bias I/O buffer length
                              1CDA  5432          ;
                              1CDA  5433          ;   Build the message
                              1CDA  5434          ;
          83    08    90    1CDA  5435          MOVB    #TR2C_MSG_NOP,(R3)+         ; Enter Phase II test msg type code
          02    01  A7    91    1CDD  5436          CMPB    ADJ$B_PTYPE(R7),#ADJ$C_PTY_PH2 ; Phase II partner?
                      1B    13  1CE1  5437          BEQL    40$                         ; If so, assumption correct
          FF  A3    05    90    1CE3  5438          MOVB    #TR3C_MSG_TST,-1(R3)        ; Partner is Phase III, replace type
                              1CE7  5439                                              ; code with Phase III test msg type code
                00    EF    1CE7  5440          EXTZV   #TR4$V_ADDR_DEST,-          ; Get our address (without area)
        50    0E  A4    0A    1CE9  5441                  #TR4$S_ADDR_DEST,RCB$W_ADDR(R4),R0
              83    50    B0  1CED  5442          MOVW    R0,(R3)+                    ; Enter source node address
              83    58    90  1CF0  5443          MOVB    R8,(R3)+                    ; Enter # of test data bytes
                    34    BB  1CF3  5444          PUSHR   #^M<R2,R4,R5>               ; Save regs
63    58    AA  8F    6E    00    2C  1CF5  5445          MOVC5   #0,(SP),#^X<AA>,R8,(R3) ; Enter test data
                    34    BA  1CFC  5446          POPR    #^M<R2,R4,R5>               ; Restore regs
                1A  A6    97  1CFE  5447 40$:     DECB    LPD$B_TSTCNT(R6)            ; Account for this test message
     0038'C2    53    C0    1D01  5448          ADDL    R3,WQE$C_LENGTH+P2(R2)      ; Setup buffer size
              50    00'   D0    1D06  5449          MOVL    S^#IO$_WRITELBLK,R0         ; Setup I/O fct code
```

```
05  1D09  5450        RSB                                  ; Return to co-routine to xmit
```

```
                     1D0A  5452          .SBTTL  ACT_ENT_RUN - Enter RUN state
                     1D0A  5453 ;+
                     1D0A  5454 ; ACT_ENT_RUN - Enter the RUN state for a non-broadcast circuit
                     1D0A  5455 ;
                     1D0A  5456 ; INPUTS:        R11     CRI CNR ptr
                     1D0A  5457 ;                R10     CRI CNF ptr
                     1D0A  5458 ;                R7      ADJ address
                     1D0A  5459 ;                R6      LPD address
                     1D0A  5460 ;                R5      WQE address
                     1D0A  5461 ;                R4      RCB address
                     1D0A  5462 ;
                     1D0A  5463 ; OUTPUTS:       R5-R7   Preserved
                     1D0A  5464 ;                R1      Next event to be processed
                     1D0A  5465 ;                R0      Low bit set if state change is permitted,
                     1D0A  5466 ;                        Low bit clear to avoid state change
                     1D0A  5467 ;
                     1D0A  5468 ;                All other regs may be clobbered.
                     1D0A  5469 ;-
                     1D0A  5470 ACT_ENT_RUN:                                 ; Enter RUN state
        04    E2     1D0A  5471          BBSS    #LPD$V_RUN,-                ;
  08 22 A6           1D0C  5472                  LPD$W_STS(R6),7$            ; Mark circuit as active for data msgs
     60 A4    96     1D0F  5473          INCB    RCB$B_ACT_DLL(R4)           ; Account for datalink
     5D A4    90     1D12  5474          MOVB    RCB$B_MAX_SNK(R4),-         ; Init square root limiter
     1E A6           1D15  5475                  LPD$B_XMT_SRL(R6)           ;
                     1D17  5476 7$:      SETBIT  #ADJ$V_RUN,ADJ$B_STS(R7) ; Mark adjacency is up
                     1D1B  5477          ;
                     1D1B  5478          ;   Start listen timer going, as long as this isn't a Phase II
                     1D1B  5479          ;   link (Phase II didn't have any mandatory hello timer).
                     1D1B  5480          ;
  02    01 A7    91    1D1B  5481          CMPB    ADJ$B_PTYPE(R7),#ADJ$C_PTY_PH2 ; If not Phase II link,
              04    13    1D1F  5482          BEQL    8$
                     1D21  5483          SETBIT  #ADJ$V_LSN,ADJ$B_STS(R7) ; Start listen timer going
                     1D25  5484 8$:      ;
                     1D25  5485          ;   If the partner node is a endnode or a Phase II node, then init
                     1D25  5486          ;   the cell in the cost/hops matrix associated with this node to
                     1D25  5487          ;   indicate that the node is directly adjacent.  This is because
                     1D25  5488          ;   we will never get any other notification (such as a routing
                     1D25  5489          ;   message) to update the cell.  The actual cost will be correctly
                     1D25  5490          ;   computed when the decision algorithm is run.
                     1D25  5491          ;
                     1D25  5492          $DISPATCH ADJ$B_PTYPE(R7),TYPE=B,<- ; Based on adjacency type
                     1D25  5493                  <ADJ$C_PTY_PH2,10$>,-   ; Phase II nodes
                     1D25  5494                  <ADJ$C_PTY_PH3N,10$>,-  ; Phase III endnodes
                     1D25  5495                  <ADJ$C_PTY_PH4N,10$>>   ; Phase IV endnodes
              35    11    1D34  5496          BRB     40$                 ; Else, skip it
                     1D36  5497
     51    02 A7    9A    1D36  5498 10$:     MOVZBL  ADJ$B_LPD_INX(R7),R1    ; Get the circuit's index
              0A    EF    1D3A  5499          EXTZV   #TR4$V_ADDR_AREA,-      ; Get area address
  52    04 A7          06    1D3C  5500                  #TR4$S_ADDR_AREA,ADJ$W_PNA(R7),R2
              16    13    1D40  5501          BEQL    30$                 ; If area = 0, assume our area
     008B C4    52    91    1D42  5502          CMPB    R2,RCB$B_HOMEAREA(R4)   ; Our area?
              0F    13    1D47  5503          BEQL    30$                 ; If so, set the right cost/hops
51    00001A88'EF41    D0    1D49  5504          MOVL    NET$AL_AREA_CH[R1],R1   ; Get address of area cost/hops buffer
              18    13    1D51  5505          BEQL    40$                 ; If none, skip it
            6142    B4    1D53  5506          CLRW    (R1)[R2]            ; Set area cost/hops to "adjacent"
              13    11    1D56  5507          BRB     40$
              00    EF    1D58  5508 30$:     EXTZV   #TR4$V_ADDR_DEST,-      ; Get node address within our area
```

```
   52   04 A7   0A        1D5A  5509                  #TR4$S_ADDR_DEST,ADJ$W_PNA(R7),R2
51   00000980'EF41   D0   1D5E  5510          MOVL    NET$AL_CH_VEC[R1],R1     ; Get address of cost/hops buffer
                 03   13   1D66  5511          BEQL    40$                      ; If none, skip it
               6142   B4   1D68  5512          CLRW    (R1)[R2]                 ; Set cost/hops word to "adjacent"
               E292'  30   1D6B  5513 40$:     BSBW    UPDATE_ALL               ; Re-run decision algorithm
                          1D6E  5514                                           ; and force routing msgs to be sent
                          1D6E  5515          ;
                          1D6E  5516          ;   Announce the circuit is up
                          1D6E  5517          ;
                          1D6E  5518          $LOG    TPL_LUP,,,R5             ; Set "circuit up" event
               E287'  30   1D76  5519          BSBW    NET$EVT_INTRAW           ; Log the event record
                          1D79  5520          ;
                          1D79  5521          ;   Start the automatic routing update timer, which causes
                          1D79  5522          ;   a routing message to be sent on this circuit each tick.
                          1D79  5523          ;
               FCCE   30   1D79  5524          BSBW    START_XRT                ; Start routing timer
           51    00   D0   1D7C  5525          MOVL    #LEV$C_NO_EVT,R1         ; No more transitions
           50    01   90   1D7F  5526          MOVB    #1,R0                    ; Allow state change
                      05   1D82  5527          RSB
```

```
                     1D83  5529          .SBTTL  ACT_BC_UP - Broadcast datalink has initialized
                     1D83  5530 ;+
                     1D83  5531 ; ACT_BC_UP     -  Start broadcast circuit Transport initialization
                     1D83  5532 ;
                     1D83  5533 ; Inputs:
                     1D83  5534 ;
                     1D83  5535 ;       R11 = CRI CNR address
                     1D83  5536 ;       R10 = CRI CNF address
                     1D83  5537 ;       R7 = ADJ address
                     1D83  5538 ;       R6 = LPD address
                     1D83  5539 ;       R5 = WQE address
                     1D83  5540 ;       R4 = RCB address
                     1D83  5541 ;
                     1D83  5542 ; Outputs:
                     1D83  5543 ;
                     1D83  5544 ;       R1 = Next event to be processed
                     1D83  5545 ;       R0 = True if state change allowed, false if not.
                     1D83  5546 ;-
                     1D83  5547 ACT_BC_UP:
        04    E2     1D83  5548          BBSS    #LPD$V_RUN,-                ;
  08 22 A6           1D85  5549                  LPD$W_STS(R6),7$            ; Mark circuit as active for data msgs
     60 A4    96     1D88  5550          INCB    RCB$B_ACT_DLL(R4)           ; Account for datalink
     5D A4    90     1D8B  5551          MOVB    RCB$B_MAX_SNK(R4),-         ; Init square root limiter
     1E A6           1D8E  5552                  LPD$B_XMT_SRL(R6)           ;
                     1D90  5553 7$:      SETBIT  ADJ$V_RTG,ADJ$B_STS(R7) ; Mark as routing adjacency
                     1D93  5554          ;
                     1D93  5555          ;   For broadcast circuits, preset the "partner buffer size" in
                     1D93  5556          ;   the main adjacency to our own buffer size.  This field will
                     1D93  5557          ;   be updated to always contain the minimum buffer size of all
                     1D93  5558          ;   the BRAs on the circuit.
                     1D93  5559          ;
      50 A6    B0     1D93  5560          MOVW    LPD$W_BUFSIZ(R6),-          ; Preset partner buffer size to our
      06 A7           1D96  5561                  ADJ$W_BUFSIZ(R7)            ; buffer size (main BC ADJ case)
                     1D98  5562          ;
                     1D98  5563          ;   Tell NETDRIVER to send a Router/Endnode Hello message immediately
                     1D98  5564          ;
   50    0D    9A     1D98  5565          MOVZBL  #NETUPD$_SEND_HELLO,R0  ; Set function code
      0F8E    30     1D9B  5566          BSBW    TELL_NETDRIVER          ; Call NETDRIVER to send hello msg
                     1D9E  5567          ;
                     1D9E  5568          ;   Re-calculate the square root limiters, to account for the
                     1D9E  5569          ;   additional circuit now active.
                     1D9E  5570          ;
      E25F'    30     1D9E  5571          BSBW    UPDATE_ALL              ; Update routing database
                     1DA1  5572          ;
                     1DA1  5573          ;   Log a "circuit up" event record.
                     1DA1  5574          ;
                     1DA1  5575          $LOG    TPL_LUP,,,R5            ; Set "circuit up" event
      E254'    30     1DA9  5576          BSBW    NET$EVT_INTRAW          ; Log the event record
                     1DAC  5577          ;
                     1DAC  5578          ;   If we are a router, start the "election suppression" timer to
                     1DAC  5579          ;   prevent our election from being resolved before we've had a chance
                     1DAC  5580          ;   to hear from everybody.
                     1DAC  5581          ;
   05    1D A6    91     1DAC  5582          CMPB    LPD$B_ETY(R6),#ADJ$C_PTY_PH4N ; Are we an endnode?
         2A    13     1DB0  5583          BEQL    20$                     ; If so, skip this
   51    20 A6    9A     1DB2  5584          MOVZBL  LPD$B_PTH_INX(R6),R1    ; Get LPD index
51 51    10    78     1DB6  5585          ASHL    #16,R1,R1               ; Shift into upper word
```

```
                51   011B 8F    B0  1DBA  5586          MOVW    #<<WQE$C_QUAL_DLL>@8>!- ; Overlay QUAL and EVT fields
                                    1DBF  5587                  LEV$C_ELECT_TIM,R1
                52   EFBD CF    9E  1DBF  5588          MOVAB   NET$DLL_PRC_WQE,R2      ; Setup action routine address
                          05    7A  1DC4  5589          EMUL    #TR$C_TIM_DRDELAY,-     ; Set timer value
          53    00    00989680 8F      1DC6  5590                  #10*1000*1000,#0,R3
                             E230' 30  1DCD  5591          BSBW    WQE$RESET_TIM           ; Set the timer
                54   00000000'EF   D0  1DD0  5592          MOVL    NET$GL_PTR_VCB,R4       ; Recover RCB address
                                       1DD7  5593          SETBIT  #LPD$V_ELECT_TIM,-      ; Mark suppression timer ticking
                                       1DD7  5594                  LPD$W_STS(R6)
                                       1DDC  5595  20$:    ;
                                       1DDC  5596          ;   Start the automatic routing update timer, which causes
                                       1DDC  5597          ;   a routing message to be sent on this circuit each tick.
                                       1DDC  5598          ;
                             FC6B      30  1DDC  5599          BSBW    START_XRT               ; Start routing timer
                                       1DDF  5600          ;
                                       1DDF  5601          ;   Notify DLE module that broadcast circuit is up, so that it
                                       1DDF  5602          ;   can enable the "load/dump" and "loopback" protocol types.
                                       1DDF  5603          ;
                             E21E'     30  1DDF  5604          BSBW    DLE$BC_UP               ; Enable service on circuit
                          51   00      D0  1DE2  5605          MOVL    #LEV$C_NO_EVT,R1        ; No more transitions
                          50   01      90  1DE5  5606          MOVB    #1,R0                   ; Allow state change
                                       05  1DE8  5607          RSB
```

```
                                1DE9  5609          .SBTTL  BRA_UP - Setup new adjacency for BRA
                                1DE9  5610 ;+
                                1DE9  5611 ; BRA_UP - Setup new adjacency control block for broadcast router
                                1DE9  5612 ;
                                1DE9  5613 ; This routine is called when a broadcast router is heard from to
                                1DE9  5614 ; allocate a new ADJ block and declare the node up.
                                1DE9  5615 ;
                                1DE9  5616 ; Inputs:
                                1DE9  5617 ;
                                1DE9  5618 ;       R6 = LPD address
                                1DE9  5619 ;       R5 = WQE address
                                1DE9  5620 ;       R4 = RCB address
                                1DE9  5621 ;       LEV_W_PNA = Address of node which sent message
                                1DE9  5622 ;       LEV_B_PRIORITY = BRA's router priority (0 if none or not available)
                                1DE9  5623 ;       PTYPE = Type of node parsed from message
                                1DE9  5624 ;
                                1DE9  5625 ; Outputs:
                                1DE9  5626 ;
                                1DE9  5627 ;       R0 = status code
                                1DE9  5628 ;       R6 = LPD address
                                1DE9  5629 ;       R7 = ADJ address
                                1DE9  5630 ;       R8 = ADJ index
                                1DE9  5631 ;
                                1DE9  5632 ;       R1-R3 are destroyed.
                                1DE9  5633 ;-
                                1DE9  5634 BRA_UP:
                                1DE9  5635          ;
                                1DE9  5636          ;   See if there is already a BRA slot for this circuit/node
                                1DE9  5637          ;   pair.  This would be the case if we received several Router
                                1DE9  5638          ;   Hello messages in a row - the first would create the BRA,
                                1DE9  5639          ;   and the subsequent messages should not create duplicate BRAs.
                                1DE9  5640          ;
          58    5C A4    9A   1DE9  5641          MOVZBL  RCB$B_MAX_LPD(R4),R8    ; Get number of circuits
          53    6A A4    3C   1DED  5642          MOVZWL  RCB$W_MAX_RTG(R4),R3    ; Set ending ADJ index
                  19    11   1DF1  5643          BRB     5$                      ; Start with slot NC+1
       57    2C B448    D0   1DF3  5644 2$:       MOVL    @RCB$L_PTR_ADJ(R4)[R8],R7 ; Get ADJ address
 04 A7  00000014'EF    B1   1DF8  5645          CMPW    LEV_W_PNA,ADJ$W_PNA(R7) ; Does the node address match?
                0A    12   1E00  5646          BNEQ    5$                      ; Branch if not
              20 A6   91   1E02  5647          CMPB    LPD$B_PTH_INX(R6),-     ; Does the circuit match?
              02 A7                1E05  5648                  ADJ$B_LPD_INX(R7)
                03    12   1E07  5649          BNEQ    5$                      ; If duplicate found,
               009D   31   1E09  5650          BRW     90$                     ; Do nothing - exit with this ADJ
          E3 58   53   F3   1E0C  5651 5$:       AOBLEQ  R3,R8,2$                ; Loop thru all BRA slots
                               1E10  5652          ;
                               1E10  5653          ;   If we are an endnode, do not allow more than 1 BRA at
                               1E10  5654          ;   a time (since the BRA is always the designated router).
                               1E10  5655          ;   As a result, if we have encountered a new BRA at this
                               1E10  5656          ;   point, bring down the old BRA with "adjancency down".
                               1E10  5657          ;
          05    1D A6    91   1E10  5658          CMPB    LPD$B_ETY(R6),#ADJ$C_PTY_PH4N   ; Are we an endnode?
                24    12   1E14  5659          BNEQ    8$                      ; Branch if not
          50    5C A4    9A   1E16  5660          MOVZBL  RCB$B_MAX_LPD(R4),R0    ; Get # circuits
          50    2C A6    B1   1E1A  5661          CMPW    LPD$W_DRT(R6),R0        ; Any external DRT active now?
                1A    15   1E1E  5662          BLEQ    8$                      ; Branch if not
              2C A6    B0   1E20  5663          MOVW    LPD$W_DRT(R6),-         ; Move DRT adjacency index to WQE
              20 A5                1E23  5664                  WQE$W_ADJ_INX(R5)
                               1E25  5665          $LOG    TPL_ARJ,,,R5            ; Setup "adjacency rejected"
```

```
                18 A5   D4   1E2D  5666          CLRL    WQE$L_EVL_PKT(R5)       ; Indicate no packet for this event
                  E1CD' 30   1E30  5667          BSBW    NET$EVT_INTRAW          ; Log the event record
             58   2C A6  3C   1E33  5668          MOVZWL  LPD$W_DRT(R6),R8        ; Get designated BRA index
                  028B   30   1E37  5669          BSBW    ADJ_DOWN                ; Bring the adjacency down
                         1E3A  5670          ;
                         1E3A  5671          ;   Allocate a new BRA adjacency slot
                         1E3A  5672          ;
             58   5C A4   9A   1E3A  5673  8$:     MOVZBL  RCB$B_MAX_LPD(R4),R8    ; Get number of circuits
             53   6A A4   3C   1E3E  5674          MOVZWL  RCB$W_MAX_RTG(R4),R3    ; Set ending ADJ index
                     09   11   1E42  5675          BRB     15$                     ; Start with slot NC+1
          57   2C B448   D0   1E44  5676  10$:     MOVL    @RCB$L_PTR_ADJ(R4)[R8],R7 ; Get ADJ address
            37 67    00   E1   1E49  5677          BBC     #ADJ$V_INUSE,ADJ$B_STS(R7),20$ ; Branch if slot available
            F3 58    53   F3   1E4D  5678  15$:     AOBLEQ  R3,R8,10$               ; Loop thru all BRA slots
                         1E51  5679          ;
                         1E51  5680          ;   The BRA database is full.  Eject the lowest priority BRA.
                         1E51  5681          ;
       52   0000001C'EF   9A   1E51  5682          MOVZBL  LEV_B_PRIORITY,R2       ; Get BRA router priority, if known
                     24   13   1E58  5683          BEQL    18$                     ; If not known, then eject newest BRA
       53   00000014'EF   3C   1E5A  5684          MOVZWL  LEV_W_PNA,R3            ; Pass newest BRA's address
                     4A   10   1E61  5685          BSBB    LOWEST_PRIO_BRA         ; Determine lowest priority BRA
                     58   D5   1E63  5686          TSTL    R8                      ; If newest BRA is lowest priority,
                     17   13   1E65  5687          BEQL    18$                     ; then simply ignore the message
             20 A5   58   B0   1E67  5688          MOVW    R8,WQE$W_ADJ_INX(R5)    ; Move adjacency index to WQE
                         1E6B  5689          $LOG    TPL_ARJ,,,R5                ; Setup "adjacency rejected"
                18 A5   D4   1E73  5690          CLRL    WQE$L_EVL_PKT(R5)       ; Indicate no packet for this event
                  E187' 30   1E76  5691          BSBW    NET$EVT_INTRAW          ; Log the event record
                  0249  30   1E79  5692          BSBW    ADJ_DOWN                ; Bring the adjacency down
                     BC   11   1E7C  5693          BRB     8$                      ; Now allocate the slot just freed up
                         1E7E  5694          ;
                         1E7E  5695          ;   The new BRA happens to be the lowest priority BRA, and thus,
                         1E7E  5696          ;   we must ignore the message we just received from it.
                         1E7E  5697          ;
          50   0000'8F   3C   1E7E  5698  18$:     MOVZWL  #SS$_INSFMEM,R0         ; Indicate BRA database full
                     05   1E83  5699          RSB                             ; and exit
                         1E84  5700          ;
                         1E84  5701          ;   ADJ slot found - initialize it
                         1E84  5702          ;
                     30   BB   1E84  5703  20$:     PUSHR   #^M<R4,R5>              ; Save registers
 67   0D   00   6E   00   2C   1E86  5704          MOVC5   #0,(SP),#0,#ADJ$C_LENGTH,(R7) ; Zero ADJ cell
                     30   BA   1E8C  5705          POPR    #^M<R4,R5>              ; Restore registers
                     88   1E8E  5706          BISB    #ADJ$M_INUSE!-          ; Mark the slot in use
                                                       ADJ$M_RTG!-             ; Mark as routing adjacency
                                                       ADJ$M_LSN,-             ; Start the listen timer going
                                                      ADJ$B_STS(R7)
                   67   0D                    
          02 A7    20 A6   B0   1E91  5710          MOVW    LPD$W_PTH(R6),ADJ$W_LPD(R7) ; Store associated LPD
    04 A7    00000014'EF   B0   1E96  5711          MOVW    LEV_W_PNA,ADJ$W_PNA(R7) ; Set partner node address
    01 A7    00000038'EF   90   1E9E  5712          MOVB    PTYPE,ADJ$B_PTYPE(R7)   ; Set partner type
                         1EA6  5713          ;
                         1EA6  5714          ;   Allocate cost/hops buffer for routing with this node
                         1EA6  5715          ;
                  E518   30   1EA6  5716          BSBW    ALLOC_COSTHOPS          ; Allocate a cost/hops buffer
                         1EA9  5717          ;
                         1EA9  5718          ;   Leave RUN flag off, until we hear a Router Hello message from
                         1EA9  5719          ;   the remote router node with our address in it, indicating that
                         1EA9  5720          ;   two-way communication has been established.  Only then will
                         1EA9  5721          ;   the router be declared up.
                         1EA9  5722          ;
```

```
                50   00'  D0  1EA9  5723 90$:    MOVL    S^#SS$_NORMAL,R0        ; Successful
                     05   1EAC  5724             RSB
```

```
                        1EAD          .SBTTL  LOWEST_PRIO_BRA - Find lowest priority BRA
                        1EAD  ;+
                        1EAD  ; LOWEST_PRIO_BRA - Determine the lowest priority BRA
                        1EAD  ;
                        1EAD  ; This routine is called when we must determine the lowest priority
                        1EAD  ; BRA in the event that the BRA database is full, and we just heard
                        1EAD  ; from another BRA.
                        1EAD  ;
                        1EAD  ; Inputs:
                        1EAD  ;
                        1EAD  ;       R6 = LPD address
                        1EAD  ;       R5 = WQE address
                        1EAD  ;       R4 = RCB address
                        1EAD  ;       R3 = Newest BRA's address
                        1EAD  ;       R2 = Newest BRA's router priority
                        1EAD  ;
                        1EAD  ; Outputs:
                        1EAD  ;
                        1EAD  ;       R8 = Lowest priority BRA, 0 if "Newest BRA" is lowest priority
                        1EAD  ;
                        1EAD  ;       R0-R3,R7 are destroyed.
                        1EAD  ;-
                        1EAD  LOWEST_PRIO_BRA:
            58    D4    1EAD          CLRL    R8                              ; Indicate no lowest ADJ yet
   57    5C A4    9A    1EAF          MOVZBL  RCB$B_MAX_LPD(R4),R7            ; Get number of circuits
   51    6A A4    3C    1EB3          MOVZWL  RCB$W_MAX_RTG(R4),R1            ; Set ending ADJ index
            22    11    1EB7          BRB     55$                             ; Start at slot NC+1
50    2C B447    D0    1EB9  50$:    MOVL    @RCB$L_PTR_ADJ(R4)[R7],R0       ; Get ADJ address
   19 60    00    E1    1EBE          BBC     #ADJ$V_INUSE,ADJ$B_STS(R0),55$  ; Skip if slot not in use
   52    0C A0    91    1EC2          CMPB    ADJ$B_BCPRI(R0),R2              ; Lower priority?
            13    1A    1EC6          BGTRU   55$                             ; Branch if not
            06    1F    1EC8          BLSSU   52$                             ; Branch if so
   53    04 A0    B1    1ECA          CMPW    ADJ$W_PNA(R0),R3                ; If equal, compare addresses
            0B    1E    1ECE          BGEQU   55$                             ; If address lower,
   53    04 A0    3C    1ED0  52$:    MOVZWL  ADJ$W_PNA(R0),R3                ; Update "lowest priority BRA"
   52    0C A0    9A    1ED4          MOVZBL  ADJ$B_BCPRI(R0),R2              ; Update "lowest priority"
      58    57    D0    1ED8          MOVL    R7,R8                           ; Update "lowest prio. index"
   DA 57    51    F3    1EDB  55$:    AOBLEQ  R1,R7,50$                       ; Loop thru all routers
                  05    1EDF          RSB
```

```
                                1EE0  5766          .SBTTL  BEA_UP - Setup new adjacency for BEA
                                1EE0  5767 ;+
                                1EE0  5768 ; BEA_UP - Setup new adjacency control block for broadcast endnode
                                1EE0  5769 ;
                                1EE0  5770 ; This routine is called when a broadcast endnode is heard from to
                                1EE0  5771 ; allocate a new ADJ block and declare the node up.
                                1EE0  5772 ;
                                1EE0  5773 ; Inputs:
                                1EE0  5774 ;
                                1EE0  5775 ;       R6 = LPD address
                                1EE0  5776 ;       LEV_W_PNA = Address of node which sent message
                                1EE0  5777 ;
                                1EE0  5778 ; Outputs:
                                1EE0  5779 ;
                                1EE0  5780 ;       R0 = status code
                                1EE0  5781 ;       R6 = LPD address
                                1EE0  5782 ;       R7 = ADJ address
                                1EE0  5783 ;       R8 = ADJ index
                                1EE0  5784 ;
                                1EE0  5785 ;       R1-R5 are destroyed.
                                1EE0  5786 ;-
                                1EE0  5787 BEA_UP:
                                1EE0  5788          ;
                                1EE0  5789          ;   See if there is already a BEA slot for this endnode.
                                1EE0  5790          ;   This would be the case if we received several Hello
                                1EE0  5791          ;   messages in a row - the first would create the BEA, and
                                1EE0  5792          ;   the subsequent messages should not create duplicate BEAs.
                                1EE0  5793          ;
          58    6A A4    3C     1EE0  5794          MOVZWL  RCB$W_MAX_RTG(R4),R8    ; Get NC + NBRA
          53    68 A4    3C     1EE4  5795          MOVZWL  RCB$W_MAX_ADJ(R4),R3    ; Set ending ADJ index
                   0F    11     1EE8  5796          BRB     5$                      ; Start with slot NC+1
       57    2C B448     D0     1EEA  5797 2$:       MOVL    @RCB$L_PTR_ADJ(R4)[R8],R7 ; Get ADJ address
 04 A7    00000014'EF    B1     1EEF  5798          CMPW    LEV_W_PNA,ADJ$W_PNA(R7) ; Does the node address match?
                   77    13     1EF7  5799          BEQL    90$                     ; If duplicate found, do nothing
          ED 58    53    F3     1EF9  5800 5$:       AOBLEQ  R3,R8,2$                ; Loop thru all BRA slots
                                1EFD  5801          ;
                                1EFD  5802          ;   Allocate a new BEA adjacency slot
                                1EFD  5803          ;
          58    6A A4    3C     1EFD  5804          MOVZWL  RCB$W_MAX_RTG(R4),R8    ; Get NC + NBRA
          53    68 A4    3C     1F01  5805          MOVZWL  RCB$W_MAX_ADJ(R4),R3    ; Set ending ADJ index
                   09    11     1F05  5806          BRB     15$                     ; Start with slot NC+NBRA+1
       57    2C B448     D0     1F07  5807 10$:      MOVL    @RCB$L_PTR_ADJ(R4)[R8],R7 ; Get ADJ address
          0A 67    00    E1     1F0C  5808          BBC     #ADJ$V_INUSE,ADJ$B_STS(R7),20$ ; Branch if slot available
          F3 58    53    F3     1F10  5809 15$:      AOBLEQ  R3,R8,10$               ; Loop thru all BEA slots
       50    0000'8F    3C     1F14  5810          MOVZWL  #SS$_INSFMEM,R0         ; Indicate BEA database full
                   05    05     1F19  5811          RSB                             ; and exit
                                1F1A  5812 20$:      ;
                                1F1A  5813          ;   ADJ slot found - initialize it
                                1F1A  5814          ;
                   30    BB     1F1A  5815          PUSHR   #^M<R4,R5>              ; Save registers
 67    0D    00    6E    00     2C     1F1C  5816          MOVC5   #0,(SP),#0,#ADJ$C_LENGTH,(R7) ; Zero ADJ cell
                   30    BA     1F22  5817          POPR    #^M<R4,R5>              ; Restore registers
                   88    88     1F24  5818          BISB    #ADJ$M_INUSE!-          ; Mark the slot in use
                                1F25  5819                   ADJ$M_RUN!-            ; Mark adjacency up for routing
                                1F25  5820                   ADJ$M_LSN,-            ; Start the listen timer going
                67    0B        1F25  5821                   ADJ$B_STS(R7)          ; and mark adjacency up for routing
       02 A7    20 A6    B0     1F27  5822          MOVW    LPD$W_PTH(R6),ADJ$W_LPD(R7) ; Store associated LPD
```

```
04 A7   00000014'EF   B0  1F2C  5823          MOVW    LEV_W_PNA,ADJ$W_PNA(R7) ; Set partner node address
         01 A7   05   90  1F34  5824          MOVB    #ADJ$C_PTY_PH4N,ADJ$B_PTYPE(R7) ; Set partner type
                          1F38  5825          ;
                          1F38  5826          ;   Set entry in cost/hops matrix for the main NI adjacency
                          1F38  5827          ;   to indicate that it is reachable (hops=0, cost=0).  The
                          1F38  5828          ;   cost/hops will be correctly computed to include this
                          1F38  5829          ;   node when the decision algorithm is run.
                          1F38  5830          ;
         51   20 A6   9A  1F38  5831          MOVZBL  LPD$B_PTH_INX(R6),R1     ; Get LPD index
                  0A   EF  1F3C  5832          EXTZV   #TR4$V_ADDR_AREA,-       ; Get area address
    52   04 A7   06       1F3E  5833                  #TR4$S_ADDR_AREA,ADJ$W_PNA(R7),R2
                  16   13  1F42  5834          BEQL    30$                      ; If area = 0, assume our area
       008B C4   52   91  1F44  5835          CMPB    R2,RCB$B_HOMEAREA(R4)    ; Our area?
                  0F   13  1F49  5836          BEQL    30$                      ; If so, set the right cost/hops
 51   00001A88'EF41   D0  1F4B  5837          MOVL    NET$AL_AREA_CH[R1],R1    ; Get address of area cost/hops buffer
                  18   13  1F53  5838          BEQL    40$                      ; If none, skip it
                6142   B4  1F55  5839          CLRW    (R1)[R2]                 ; Set area cost/hops to "adjacent"
                  13   11  1F58  5840          BRB     40$
                  00   EF  1F5A  5841 30$:     EXTZV   #TR4$V_ADDR_DEST,-       ; Get node address within our area
    52   04 A7   0A       1F5C  5842                  #TR4$S_ADDR_DEST,ADJ$W_PNA(R7),R2
 51   00000980'EF41   D0  1F60  5843          MOVL    NET$AL_CH_VEC[R1],R1     ; Get address of cost/hops buffer
                  03   13  1F68  5844          BEQL    40$                      ; If none, skip it
                6142   B4  1F6A  5845          CLRW    (R1)[R2]                 ; Set cost/hops word to "adjacent"
                          1F6D  5846 40$:     ;
                          1F6D  5847          ;   Update the routing database
                          1F6D  5848          ;
                E090'  30  1F6D  5849          BSBW    UPDATE_ALL               ; Re-run decision algorithm
                          1F70  5850                                           ; and force routing msgs to be sent
            50   00'   D0  1F70  5851 90$:     MOVL    S^#SS$_NORMAL,R0         ; Successful
                      05  1F73  5852          RSB
```

```
                     1F74  5854          .SBTTL  Error action routines for "RUN" state
                     1F74  5855 ;+
                     1F74  5856 ; ACT_RUN_SYNC -   Synchronization lost while in the "run" state
                     1F74  5857 ; ACT_RUN_UXPK -   Unexpected packet type while in the "run" state
                     1F74  5858 ; ACT_ENT_MPR  -   Circuit has entered MOP mode while in the "run" state
                     1F74  5859 ; ACT_RUN_SHUT -   Shut down the datalink while in the "run" state
                     1F74  5860 ;
                     1F74  5861 ; INPUTS:        R11     CRI CNR ptr
                     1F74  5862 ;                R10     CRI CNF ptr
                     1F74  5863 ;                R6      LPD ptr
                     1F74  5864 ;                R5      WQE address
                     1F74  5865 ;                R4      RCB address
                     1F74  5866 ;
                     1F74  5867 ; OUTPUTS:       R5      Unchanged
                     1F74  5868 ;                R1      Next event to be processed
                     1F74  5869 ;                R0      Low bit set if state change is permitted,
                     1F74  5870 ;                        Low bit clear to avoid state change
                     1F74  5871 ;
                     1F74  5872 ;                All other regs may be clobbered.
                     1F74  5873 ;-
                     1F74  5874 ACT_RUN_SYNC:                            ; Circuit down - synchronization lost
                     1F74  5875          $LOG    TPL_LDF,TPL_PRSN_SYNC,,R5 ; Setup logging data
51   23   D0   1F7C  5876          MOVL    #LEV$C_LOG_CDE,R1           ; Signal "circuit down" event
     50   D4   1F7F  5877          CLRL    R0                          ; Do not change state for this event
          05   1F81  5878          RSB
                     1F82  5879
                     1F82  5880 ACT_RUN_UXPK:                           ; Circuit down - unexpected packet type
                     1F82  5881          $LOG    TPL_LDS,TPL_PRSN_UXPK,,R5 ; Setup logging data
51   24   D0   1F8A  5882          MOVL    #LEV$C_LOG_ADE,R1           ; Signal "adjacency down" event
     50   D4   1F8D  5883          CLRL    R0                          ; Do not change state for this event
          05   1F8F  5884          RSB
                     1F90  5885
                     1F90  5886 ACT_ENT_MPR:                            ; Enter MOP mode from the run state
          36   10   1F90  5887          BSBB    EXIT_RUN_STATE          ; Exit the "run" state
        02E6   30   1F92  5888          BSBW    ACT_DIO_SHUT            ; Shutdown the circuit
51      21   D0   1F95  5889          MOVL    #LEV$C_IRP_MM,R1        ; Resignal MOP mode event
50      01   D0   1F98  5890          MOVL    #1,R0                   ; Allow state change
          05   1F9B  5891          RSB
                     1F9C  5892
                     1F9C  5893 ACT_RUN_SHUT:
          2A   10   1F9C  5894          BSBB    EXIT_RUN_STATE          ; Exit the "run" state
51      04   D0   1F9E  5895          MOVL    #LEV$C_REQ_SHUT,R1      ; Chain to "request shutdown" event
50      01   D0   1FA1  5896          MOVL    #1,R0                   ; Allow state change
          05   1FA4  5897          RSB
                     1FA5  5898
                     1FA5  5899 ACT_ADJ_DOWN:
                     1FA5  5900          ;
                     1FA5  5901          ;   If this is a non-broadcast circuit, or the adjacency is the
                     1FA5  5902          ;   primary circuit adjacency, then bring down the entire circuit.
                     1FA5  5903          ;   Otherwise, mark the adjacency down, and leave the circuit running.
                     1FA5  5904          ;
18 22 A6    0A   E1 1FA5  5905          BBC     #LPD$V_BC,LPD$W_STS(R6),50$ ; Branch if non-broadcast circuit
   50   5C A4   9A 1FAA  5906          MOVZBL  RCB$B_MAX_LPD(R4),R0    ; Get number of circuits
   50   20 A5   B1 1FAE  5907          CMPW    WQE$W_ADJ_INX(R5),R0    ; Is it the main circuit adjacency?
          0E   1B 1FB2  5908          BLEQU   50$                     ; If so, shutdown entire circuit
   58   20 A5   3C 1FB4  5909          MOVZWL  WQE$W_ADJ_INX(R5),R8    ; Get ADJ index
        010A   30 1FB8  5910          BSBW    ADJ_DOWN                ; Mark adjacency down
```

```
                51   00   D0  1FBB  5911          MOVL    #LEV$C_NO_EVT,R1        ; Nothing more to do
                50   01   D0  1FBE  5912          MOVL    #1,R0
                          05  1FC1  5913          RSB
                              1FC2  5914          ;
                              1FC2  5915          ;   Shutdown the entire circuit
                              1FC2  5916          ;
                51   11   D0  1FC2  5917  50$:    MOVL    #LEV$C_LIN_DOWN,R1      ; Chain to bring down entire circuit
                     50   D4  1FC5  5918          CLRL    R0                      ; Do not change state
                          05  1FC7  5919          RSB
```

```
                                1FC8  5921          .SBTTL  EXIT_RUN_STATE - Exit the RUN state
                                1FC8  5922 ;+
                                1FC8  5923 ; EXIT_RUN_STATE - Perform any cleanup before exiting the "run" state
                                1FC8  5924 ;
                                1FC8  5925 ; Inputs:
                                1FC8  5926 ;
                                1FC8  5927 ;       R11 = CRI CNR address
                                1FC8  5928 ;       R10 = CRI CNF address
                                1FC8  5929 ;       R7 = ADJ address
                                1FC8  5930 ;       R6 = LPD address
                                1FC8  5931 ;       R4 = RCB address
                                1FC8  5932 ;
                                1FC8  5933 ; Outputs:
                                1FC8  5934 ;
                                1FC8  5935 ;       None
                                1FC8  5936 ;-
                                1FC8  5937 EXIT_RUN_STATE:
                                1FC8  5938          BUMP    B,LPD$B_CNT_LDN(R6)     ; Increment circuit down count
                          04    E5  1FD1  5939          BBCC    #LPD$V_RUN,-            ; If leaving run state then
                    03 22 A6        1FD3  5940                  LPD$W_STS(R6),7$        ;
                       60 A4    97  1FD6  5941          DECB    RCB$B_ACT_DLL(R4)       ; Account for loss of datalink
                                1FD9  5942 7$:          ;
                                1FD9  5943          ;   Mark as unreachable all nodes which were to use this path
                                1FD9  5944          ;
                     53   20 A6    9A  1FD9  5945          MOVZBL  LPD$B_PTH_INX(R6),R3    ; Get index of LPD now inactive
                     52   5A A4    3C  1FDD  5946          MOVZWL  RCB$W_MAX_ADDR(R4),R2   ; Get maximum number of nodes
                        51   01    D0  1FE1  5947          MOVL    #1,R1                   ; Start at node #1
                  50   1C B441    3C  1FE4  5948 10$:     MOVZWL  @RCB$L_PTR_OA(R4)[R1],R0 ; Get output ADJ for this node
                           0F     13  1FE9  5949          BEQL    20$                     ; Branch if none
                  50   2C B440    D0  1FEB  5950          MOVL    @RCB$L_PTR_ADJ(R4)[R0],R0 ; Get ADJ address
                     53   02 A0    B1  1FF0  5951          CMPW    ADJ$B_LPD_INX(R0),R3    ; Does this ADJ use the LPD?
                           04     12  1FF4  5952          BNEQ    20$                     ; Branch if not
                        1C B441   B4  1FF6  5953          CLRW    @RCB$L_PTR_OA(R4)[R1]   ; Mark node unreachable
                  E6 51    52    F3  1FFA  5954 20$:     AOBLEQ  R2,R1,T0$               ; Loop through entire OA vector
                                1FFE  5955          ;
                                1FFE  5956          ;   Mark as unreachable all nodes which were to use this path
                                1FFE  5957          ;
                              20 A4   D5  1FFE  5958          TSTL    RCB$L_PTR_AOA(R4)       ; Are we an area router?
                                 22   13  2001  5959          BEQL    25$                     ; If not, skip it
                  52   008C C4    9A  2003  5960          MOVZBL  RCB$B_MAX_AREA(R4),R2   ; Get maximum number of areas
                     51    01    D0  2008  5961          MOVL    #1,R1                   ; Start at area #1
                  50   20 B441   3C  200B  5962 22$:     MOVZWL  @RCB$L_PTR_AOA(R4)[R1],R0 ; Get output ADJ for this area
                           0F     13  2010  5963          BEQL    24$                     ; Branch if none
                  50   2C B440    D0  2012  5964          MOVL    @RCB$L_PTR_ADJ(R4)[R0],R0 ; Get ADJ address
                     53   02 A0    B1  2017  5965          CMPW    ADJ$B_LPD_INX(R0),R3    ; Does this ADJ use the LPD?
                           04     12  201B  5966          BNEQ    24$                     ; Branch if not
                        20 B441   B4  201D  5967          CLRW    @RCB$L_PTR_AOA(R4)[R1]  ; Mark area unreachable
                  E6 51    52    F3  2021  5968 24$:     AOBLEQ  R2,R1,22$               ; Loop through entire AOA vector
                                2025  5969 25$:          ;
                                2025  5970          ;   Bring down the adjacency which initiated this event
                                2025  5971          ;
                     58   20 A5    3C  2025  5972          MOVZWL  WQE$W_ADJ_INX(R5),R8    ; Set index of ADJ we have in-hand
                           0099    30  2029  5973          BSBW    ADJ_DOWN                ; Bring down adjacency
                                202C  5974          ;
                                202C  5975          ;   If this is a broadcast circuit, then bring down any
                                202C  5976          ;   adjacencies that are associated with this circuit.
                                202C  5977          ;
```

```
                34 22 A6    0A   E1  202C  5978          BBC     #LPD$V_BC,LPD$W_STS(R6),50$ ; If non-broadcast circuit, we're done
                   52       68 A4 3C  2031  5979          MOVZWL  RCB$W_MAX_ADJ(R4),R2    ; Get number of adjacencies
                      58    01   D0  2035  5980          MOVL    #1,R8                   ; Start at ADJ #1
                    50    2C B448  D0  2038  5981 30$:    MOVL    @RCB$L_PTR_ADJ(R4)[R8],R0 ; Get ADJ address
                      0A 60    00   E1  203D  5982          BBC     #ADJ$V_INUSE,ADJ$B_STS(R0),40$ ; Branch if slot not in use
                20 A6    02 A0    91  2041  5983          CMPB    ADJ$B_LPD_INX(R0),LPD$B_PTH_INX(R6) ; Does it point to LPD?
                            03    12  2046  5984          BNEQ    40$                     ; If not, go on
                         007A     30  2048  5985          BSBW    ADJ_DOWN                ; Bring down the adjacency
                   E9 58    52   F3  204B  5986 40$:     AOBLEQ  R2,R8,30$               ; Loop thru entire ADJ vector
                                     204F  5987          ;
                                     204F  5988          ;   Make sure the "designated router" is reset when the
                                     204F  5989          ;   circuit is brought down, just in case it fails to get
                                     204F  5990          ;   reset properly elsewhere.
                                     204F  5991          ;
                      20 A6    9B  204F  5992          MOVZBW  LPD$B_PTH_INX(R6),-     ; Indicate no "known" designated router
                      2C A6        2052  5993                  LPD$W_DRT(R6)           ; (make it the circuit itself)
                                     2054  5994          ;
                                     2054  5995          ;   Reset router/state list to a null string, since there are
                                     2054  5996          ;   no longer any BRAs for this adjacency.
                                     2054  5997          ;
                   50    2E A6    D0  2054  5998          MOVL    LPD$L_RTR_LIST(R6),R0   ; Get address of election list
                            08    13  2058  5999          BEQL    45$                     ; Skip if none
                            60    94  205A  6000          CLRB    (R0)                    ; Reset election list to null
                                     205C  6001          ;
                                     205C  6002          ;   Send an "I'm going away" message (empty RHEL), if possible,
                                     205C  6003          ;   to notify other nodes that we are going away.
                                     205C  6004          ;
                      50    0D   9A  205C  6005          MOVZBL  #NETUPD$_SEND_HELLO,R0  ; Set function code
                         0CCA     30  205F  6006          BSBW    TELL_NETDRIVER          ; Call NETDRIVER to send hello msg
                                     2062  6007 45$:     ;
                                     2062  6008          ;   Notify DLE module that broadcast circuit is down, so that it
                                     2062  6009          ;   can disable the "load/dump" and "loopback" protocol types.
                                     2062  6010          ;
                            DF9B'    30  2062  6011          BSBW    DLE$BC_DOWN             ; Disable service on circuit
                                     2065  6012          ;
                                     2065  6013          ;   Store infinite cost/hops for all nodes in the cost/hops buffer
                                     2065  6014          ;   associated with the circuit going down.  When the decision
                                     2065  6015          ;   algorithm is run again, the cost/hops to all nodes will be
                                     2065  6016          ;   re-computed.
                                     2065  6017          ;
                      53    20 A6    9A  2065  6018 50$:     MOVZBL  LPD$B_PTH_INX(R6),R3    ; Get index of LPD now inactive
                                  38  BB  2069  6019          PUSHR   #^M<R3,R4,R5>           ; Save critical regs
                50    00000980'EF43  D0  206B  6020          MOVL    NET$AL_CH_VEC[R3],R0    ; Get address of cost/hops buffer
                                  0C  13  2073  6021          BEQL    52$                     ; Skip if none
                   FF 8F    6E    00  2C  2075  6022          MOVC5   #0,(SP),#-1,-           ; Store infinity in each cell for
                      60    0800 8F      207A  6023                  #2*NUM_NODES,(R0)       ;  each node as known to this circuit
                            53    6E  D0  207E  6024          MOVL    (SP),R3                 ; Recover R3
                50    00001A88'EF43  D0  2081  6025 52$:     MOVL    NET$AL_AREA_CH[R3],R0   ; Get address of cost/hops buffer
                                  09  13  2089  6026          BEQL    55$                     ; Skip if none
                   FF 8F    6E    00  2C  208B  6027          MOVC5   #0,(SP),#-1,-           ; Store infinity in each cell for
                      60    0080 8F      2090  6028                  #2*NUM_AREAS,(R0)       ;  each node as known to this circuit
                                  38  BA  2094  6029 55$:     POPR    #^M<R3,R4,R5>           ; Restore regs
                                     2096  6030          ;
                                     2096  6031          ;   When exiting "run" state for any reason on a X.25 PVC (including
                                     2096  6032          ;   because the remote side sent us a reset), issue a reset to the
                                     2096  6033          ;   remote side to ensure that it restarts the initialization sequence.
                                     2096  6034          ;   We can't get into an infinite loop doing this, because its only
```

```
                                                J 13
                     2096  6035          ;   done when exiting the run state.
                     2096  6036          ;
          07   E1    2096  6037          BBC     #LPD$V_X25,-            ; If X.25 circuit,
      26 22 A6         2098  6038                  LPD$W_STS(R6),60$
                     209B  6039          $GETFLD cri,l,use               ; Get circuit usage
        16 50   E9   20A8  6040          BLBC    R0,60$
       00   58   D1   20AB  6041          CMPL    R8,#NMA$C_CIRUS_PER     ; X.25 PVC?
            11   12   20AE  6042          BNEQ    60$                     ; If so,
            51   D4   20B0  6043          CLRL    R1                      ; No QIO buffer needed
          0A1C   30   20B2  6044          BSBW    NET$DLL_QIO_CO          ; Allocate and init WQE (co-routine)
0030'C2   03   D0   20B5  6045          MOVL    #PSI$C_RESET,WQE$C_LENGTH+P4(R2) ; Set P4 to "reset initiate"
50   0000'8F   3C   20BA  6046          MOVZWL  #IO$_NETCONTROL,R0      ; Set I/O function code
            9E   16   20BF  6047          JSB     @(SP)+                  ; Issue I/O request
                     20C1  6048 60$:     ;
                     20C1  6049          ;   Update the routing data base to account for the decrease
                     20C1  6050          ;   in active circuits, as well as to remove all least cost
                     20C1  6051          ;   paths over this circuit.
                     20C1  6052          ;
         DF3C'   30  20C1  6053          BSBW    UPDATE_ALL              ; Re-run decision algorithm
                     20C4  6054                                          ; and force routing msgs to be sent
                05   20C4  6055          RSB
```

```
                                20C5  6057          .SBTTL  ADJ_DOWN - Mark adjacency as shutdown
                                20C5  6058 ;+
                                20C5  6059 ; ADJ_DOWN - Shutdown the adjacency
                                20C5  6060 ;
                                20C5  6061 ; This routine is called to mark an adjacency as shutdown.
                                20C5  6062 ;
                                20C5  6063 ; Inputs:
                                20C5  6064 ;
                                20C5  6065 ;       R8 = ADJ index
                                20C5  6066 ;       R6 = LPD address
                                20C5  6067 ;       R4 = RCB address
                                20C5  6068 ;
                                20C5  6069 ; Outputs:
                                20C5  6070 ;
                                20C5  6071 ;       None
                                20C5  6072 ;
                                20C5  6073 ;       No registers are destroyed.
                                20C5  6074 ;-
                                20C5  6075 ADJ_DOWN:
            008E 8F     BB  20C5  6076          PUSHR   #^M<R1,R2,R3,R7>        ; Save registers
        57  2C B448     D0  20C9  6077          MOVL    @RCB$L_PTR_ADJ(R4)[R8],R7 ; Get ADJ address
                00      EF  20CE  6078          EXTZV   #TR4$V_ADDR_DEST,-      ; Save node # for later in routine
    53  04 A7   0A         20D0  6079                  #TR4$S_ADDR_DEST,ADJ$W_PNA(R7),R3 ; (assume it's in our area)
              04 A7     B4  20D4  6080          CLRW    ADJ$W_PNA(R7)           ; Adjacent node is unknown
              06 A7     B4  20D7  6081          CLRW    ADJ$W_BUFSIZ(R7)        ; Reset buffer size
                0E      8A  20DA  6082          BICB    #ADJ$M_RUN!ADJ$M_LSN!ADJ$M_RTG,- ; Clear flags
                67          20DC  6083                  ADJ$B_STS(R7)
              FF 8F     90  20DD  6084          MOVB    #ADJ$C_PTY_UNK,-        ; Mark partner type unknown
              01 A7         20E0  6085                  ADJ$B_PTYPE(R7)
                            20E2  6086          ;
                            20E2  6087          ;   If this is the main circuit adjacency, then do nothing more
                            20E2  6088          ;   then resetting the fields in the ADJ.
                            20E2  6089          ;
        50   5C A4     9A  20E2  6090          MOVZBL  RCB$B_MAX_LPD(R4),R0    ; Get number of circuits
           50   58     B1  20E6  6091          CMPW    R8,R0                   ; Is this the main circuit adajcency?
                26     1B  20E9  6092          BLEQU   90$                     ; If so, don't ever deallocate it
                00     E5  20EB  6093          BBCC    #ADJ$V_INUSE,-          ; Mark slot no longer in use
             22 67          20ED  6094                  ADJ$B_STS(R7),90$       ; and exit if already was marked down
                57     D4  20EF  6095          CLRL    R7                      ; Invalidate pointer
        6A A4   58     B1  20F1  6096          CMPW    R8,RCB$W_MAX_RTG(R4)    ; BRA or BEA?
                05     14  20F5  6097          BGTR    30$                     ; Branch if endnode
                           20F7  6098          ;
                           20F7  6099          ;   If this is a broadcast router, then call another routine
                           20F7  6100          ;   to handle it.
                           20F7  6101          ;
              001F     30  20F7  6102          BSBW    BRA_DOWN                ; Mark BRA down
                12     11  20FA  6103          BRB     50$
                           20FC  6104          ;
                           20FC  6105          ;   If this is an endnode, then set the cost/hops to this node
                           20FC  6106          ;   to infinity.
                           20FC  6107          ;
        50   20 A6     9A  20FC  6108 30$:     MOVZBL  LPD$B_PTH_INX(R6),R0    ; Get LPD index
50  00000980'EF40     D0  2100  6109          MOVL    NET$AL_CH_VEC[R0],R0    ; Get address of cost/hops buffer
                04     13  2108  6110          BEQL    50$                     ; If none, skip it
          6043  01     AE  210A  6111          MNEGW   #1,(R0)[R3]             ; Set cost/hops to infinity
                           210E  6112          ;
                           210E  6113          ;   Update the routing database to account for the change in
```

```
                     210E  6114                ;   the cost/hops matrix.
                     210E  6115                ;
   DEEF'  30         210E  6116 50$:    BSBW    UPDATE_ALL                ; Re-run decision algorithm
                     2111  6117                                          ; and force routing msgs to be sent
50   01   D0         2111  6118 90$:    MOVL    #1,R0                     ; Success
008E 8F   BA         2114  6119         POPR    #^M<R1,R2,R3,R7>          ; Restore registers
          05         2118  6120         RSB
```

```
                                2119  6122          .SBTTL  BRA_DOWN - Mark BRA down
                                2119  6123 ;+
                                2119  6124 ; BRA_DOWN - Mark BRA down
                                2119  6125 ;
                                2119  6126 ; This routine is called when a BRA is removed from the adjacency database.
                                2119  6127 ;
                                2119  6128 ; Inputs:
                                2119  6129 ;
                                2119  6130 ;       R8 = ADJ index
                                2119  6131 ;       R6 = LPD address
                                2119  6132 ;       R4 = RCB address
                                2119  6133 ;
                                2119  6134 ; Outputs:
                                2119  6135 ;
                                2119  6136 ;       None
                                2119  6137 ;
                                2119  6138 ;       R0-R3 are destroyed.
                                2119  6139 ;-
                                2119  6140 BRA_DOWN:
                                2119  6141           ;
                                2119  6142           ;   If this BRA was the designated router, then reset the ADJ index
                                2119  6143           ;   for the designated router, indicating no designated router.
                                2119  6144           ;
  2C A6    58    B1             2119  6145           CMPW    R8,LPD$W_DRT(R6)        ; Is the designated router going down?
        05    12                211D  6146           BNEQ    5$                      ; Branch if not
     20 A6    9B                211F  6147           MOVZBW  LPD$B_PTH_INX(R6),-     ; Indicate no "known" designated router
     2C A6                      2122  6148                   LPD$W_DRT(R6)           ; (make it the NI itself)
                                2124  6149 5$:       ;
                                2124  6150           ;   For broadcast routers, remove its entry from our Router
                                2124  6151           ;   Hello NI LIST, and set cost/hops to all nodes thru this
                                2124  6152           ;   router to infinity (by deallocating the cost/hops buffer
                                2124  6153           ;   for this router adjacency slot).
                                2124  6154           ;
                                2124  6155           $DISPATCH LPD$B_ETY(R6),TYPE=B,<- ; If we are an endnode,
                                2124  6156                   <ADJ$C_PTY_PH4N,10$>,-  ; skip the following
                                2124  6157                   <ADJ$C_PTY_PH3N,10$>>
        0067    30              2133  6158           BSBW    BUILD_RTR_LIST          ; Re-build RTR_LIST, minus this router
50 00000980'EF48    D0          2136  6159           MOVL    NET$AL_CH_VEC[R8],R0    ; Get cost/hops buffer
          0D    13              213E  6160           BEQL    10$                     ; Skip if not there
        50  0C    C2            2140  6161           SUBL    #12,R0                  ; Get address of real buffer
     DEBA'      30              2143  6162           BSBW    NET$DEALLOCATE          ; Deallocate it
00000980'EF48    D4            2146  6163           CLRL    NET$AL_CH_VEC[R8]       ; Indicate buffer no longer present
50 00001A88'EF48    D0          214D  6164 10$:     MOVL    NET$AL_AREA_CH[R8],R0   ; Get area cost/hops buffer
          0D    13              2155  6165           BEQL    15$                     ; Skip if not there
        50  0C    C2            2157  6166           SUBL    #12,R0                  ; Get address of real buffer
     DEA3'      30              215A  6167           BSBW    NET$DEALLOCATE          ; Deallocate it
00001A88'EF48    D4            215D  6168           CLRL    NET$AL_AREA_CH[R8]      ; Indicate buffer no longer present
                                2164  6169 15$:      ;
                                2164  6170           ;   Re-calculate the "minimum blocksize of all BRAs on the NI"
                                2164  6171           ;   (which is stored in the main adjacency of the NI, and is used
                                2164  6172           ;   to determine the size of routing messages sent over the NI)
                                2164  6173           ;   by scanning all active BRAs left, and determining the minimum.
                                2164  6174           ;
  50    20 A6    9A             2164  6175           MOVZBL  LPD$B_PTH_INX(R6),R0    ; Get LPD index
50   2C B440    D0              2168  6176           MOVL    @RCB$C_PTR_ADJ(R4)[R0],R0 ; Get main ADJ address for BC
        50 A6    B0            216D  6177           MOVW    LPD$W_BUFSIZ(R6),-      ; Preset minimum to our size
        06 A0                  2170  6178                   ADJ$W_BUFSIZ(R0)
```

```
                52    5C A4    9A  2172  6179          MOVZBL  RCB$B_MAX_LPD(R4),R2     ; Get number of circuits
                53    6A A4    3C  2176  6180          MOVZWL  RCB$W_MAX_RTG(R4),R3     ; Set ending ADJ index
                         1C    11  217A  6181          BRB     25$                      ; Start with slot NC+1
            51    2C B442    D0  217C  6182 20$:       MOVL    @RCB$L_PTR_ADJ(R4)[R2],R1 ; Get ADJ address
              13 61    01    E1  2181  6183          BBC     #ADJ$V_RUN,ADJ$B_STS(R1),25$ ; Skip if BRA not running
                    20 A6    91  2185  6184          CMPB    LPD$B_PTH_INX(R6),-      ; Is it on this NI?
                    02 A1            2188  6185                  ADJ$B_LPD_INX(R1)
                       0C    12  218A  6186          BNEQ    25$                      ; If not, skip it
                    06 A1    B1  218C  6187          CMPW    ADJ$W_BUFSIZ(R1),-       ; Buffer size less than minimum?
                    06 A0            218F  6188                  ADJ$W_BUFSIZ(R0)
                       05    1E  2191  6189          BGEQU   25$                      ; Branch if not
                    06 A1    B0  2193  6190          MOVW    ADJ$W_BUFSIZ(R1),-       ; If so, store the minimum
                    06 A0            2196  6191                  ADJ$W_BUFSIZ(R0)
            E0 52    53    F3  2198  6192 25$:       AOBLEQ  R3,R2,20$                ; Loop thru all BRA slots
                           05  219C  6193          RSB
```

```
                                219D  6195          .SBTTL  BUILD_RTR_LIST - Re-build NI router/state list
                                219D  6196 ;+
                                219D  6197 ; BUILD_RTR_LIST - Re-build NI router/state list
                                219D  6198 ;
                                219D  6199 ; This routine is called when the router adjacency database changes,
                                219D  6200 ; to rebuild the NI router/state list for our Router Hello message.
                                219D  6201 ;
                                219D  6202 ; Inputs:
                                219D  6203 ;
                                219D  6204 ;       R6 = LPD address
                                219D  6205 ;       R4 = RCB address
                                219D  6206 ;
                                219D  6207 ; Outputs:
                                219D  6208 ;
                                219D  6209 ;       R0 = True if election list has not changed since the last one
                                219D  6210 ;            ("election stablized").  False if it's different than
                                219D  6211 ;            the last one (send it out now).
                                219D  6212 ;
                                219D  6213 ;       R1-R3 are destroyed.
                                219D  6214 ;-
                                219D  6215 BUILD_RTR_LIST:
          0180 8F   BB          219D  6216          PUSHR   #^M<R7,R8>                      ; Save registers
     5E 000000ED 8F   C2          21A1  6217          SUBL    #1+TR4C_MAX_RSLIST,SP           ; Allocate buffer on stack
          53   5E   D0          21A8  6218          MOVL    SP,R3                           ; Point to buffer
                                21AB  6219          ;
                                21AB  6220          ;   Scan all BRA adjacencies, and for each slot in use on this LPD,
                                21AB  6221          ;   store an entry in the list.
                                21AB  6222          ;
          58   5C A4   9A      21AB  6223          MOVZBL  RCB$B_MAX_LPD(R4),R8            ; Get number of circuits
                    58   D6      21AF  6224          INCL    R8                              ; Set starting ADJ index
          51   6A A4   3C      21B1  6225          MOVZWL  RCB$W_MAX_RTG(R4),R1            ; Set ending ADJ index
     57   2C B448   D0          21B5  6226 50$:     MOVL    @RCB$[_PTR_ADJ(R4)[R8],R7       ; Get ADJ address
        24 67   00   E1          21BA  6227          BBC     #ADJ$V_INUSE,ADJ$B_STS(R7),55$  ; Skip if slot not in use
                20 A6   91      21BE  6228          CMPB    LPD$B_PTH_INX(R6),-             ; Is it on this NI?
                02 A7           21C1  6229                  ADJ$B_LPD_INX(R7)
                    1D   12      21C3  6230          BNEQ    55$                             ; If not, skip it
     83 000400AA 8F   D0          21C5  6231          MOVL    #TR$C_NI_PREFIX,(R3)+           ; Store standard Phase IV prefix
          83   04 A7   B0      21CC  6232          MOVW    ADJ$W_PNA(R7),(R3)+             ; Store node address
                                21D0  6233          ASSUME  TR4V_RS_PRIO EQ 0
          83   0C A7   90      21D0  6234          MOVB    ADJ$B_BCPRI(R7),(R3)+           ; Store router priority
                                21D4  6235          SETBIT  TR4V_RS_TWOWAY,-1(R3)           ; Assume two-way established
          05 67   01   E0      21D9  6236          BBS     #ADJ$V_RUN,ADJ$B_STS(R7),55$    ; Branch if two-way
                                21DD  6237          CLRBIT  TR4V_RS_TWOWAY,-1(R3)           ; Else, clear two-way flag
          CF 58   51   F3      21E2  6238 55$:     AOBLEQ  R1,R8,50$                       ; Loop thru all routers
               53   5E   C2      21E6  6239          SUBL    SP,R3                           ; Compute size of new list
                                21E9  6240          ;
                                21E9  6241          ;   See if new election list is different than our old one.  If
                                21E9  6242          ;   so, set a flag for the caller.
                                21E9  6243          ;
                    3E   BB      21E9  6244          PUSHR   #^M<R1,R2,R3,R4,R5>             ; Save registers
          51   2E A6   D0      21EB  6245          MOVL    LPD$L_RTR_LIST(R6),R1           ; Get address of buffer
               50   81   9A      21EF  6246          MOVZBL  (R1)+,R0                        ; Get size of current list
61   50   00   14 AE   53   2D   21F2  6247          CMPC5   R3,5*4(SP),#0,R0,(R1)           ; Is the new list different?
                    04   13      21F9  6248          BEQL    57$                             ; If so,
                    50   D4      21FB  6249          CLRL    R0                              ; "Election not stable yet"
                    03   11      21FD  6250          BRB     59$
               50   01   D0      21FF  6251 57$:     MOVL    #1,R0                           ; "Election stablized"
```

```
                     3E   BA   2202  6252 59$:    POPR    #^M<R1,R2,R3,R4,R5>              ; Restore registers
                                2204  6253         ;
                                2204  6254         ;   Store new list in LPD buffer
                                2204  6255         ;
              51   2E A6   D0   2204  6256         MOVL    LPD$L_RTR_LIST(R6),R1            ; Get address of buffer
                 81   53   90   2208  6257         MOVB    R3,(R1)+                         ; Store size of list
                      31   BB   220B  6258         PUSHR   #^M<R0,R4,R5>                    ; Save registers
           61   0C AE   53   28   220D  6259         MOVC    R3,3*4(SP),(R1)                  ; Store entire list
                      31   BA   2212  6260         POPR    #^M<R0,R4,R5>                    ; Restore registers
        5E   000000ED 8F   C0   2214  6261         ADDL    #1+TR4C_MAX_RSLIST,SP            ; Deallocate buffer on stack
                 0180 8F   BA   221B  6262         POPR    #^M<R7,R8>                       ; Restore registers
                           05   221F  6263         RSB
```

```
                                2220  6265          .SBTTL  ELECT_ROUTER - Elect designated router
                                2220  6266 ;+
                                2220  6267 ; ELECT_ROUTER - Elect the "designated router" for this circuit
                                2220  6268 ;
                                2220  6269 ; This routine elects the designated router from among all the routers
                                2220  6270 ; on this NI.  Since every router uses the same algorithm to decide,
                                2220  6271 ; all the routers arrive at the same conclusion without consultation.
                                2220  6272 ; This routine must only be called if we are a router (if an endnode
                                2220  6273 ; was to set its DRT to ourself, we would probably crash).
                                2220  6274 ;
                                2220  6275 ; Inputs:
                                2220  6276 ;
                                2220  6277 ;       R6 = LPD address
                                2220  6278 ;       R4 = RCB address
                                2220  6279 ;
                                2220  6280 ; Outputs:
                                2220  6281 ;
                                2220  6282 ;       R1 = Adjacency index of "designated router"
                                2220  6283 ;       R2 = Priority of "designated router"
                                2220  6284 ;       R3 = Node address of "designated router"
                                2220  6285 ;
                                2220  6286 ;       R0 is destroyed.
                                2220  6287 ;-
                                2220  6288 ELECT_ROUTER:
      0380 8F    BB            2220  6289          PUSHR   #^M<R7,R8,R9>                   ; Save registers
59    008B C4    9A            2224  6290          MOVZBL  RCB$B_HOMEAREA(R4),R9           ; Get our own area number
      58   01    D0            2229  6291          MOVL    #LPD$C_LOC_INX,R8               ; Set "current DRT adj index"
  52    2A A6    9A            222C  6292          MOVZBL  LPD$B_BCPRI(R6),R2              ; Set "highest priority"
  53    0E A4    3C            2230  6293          MOVZWL  RCB$W_ADDR(R4),R3               ; Set "current DRT address"
  57    5C A4    9A            2234  6294          MOVZBL  RCB$B_MAX_LPD(R4),R7            ; Get number of circuits
  51    6A A4    3C            2238  6295          MOVZWL  RCB$W_MAX_RTG(R4),R1            ; Set ending ADJ index
              31    11        223C  6296          BRB     55$                             ; Start at slot NC+1
50    2C B447    D0            223E  6297 50$:     MOVL    @RCB$L_PTR_ADJ(R4)[R7],R0       ; Get ADJ address
  28 60    00    E1            2243  6298          BBC     #ADJ$V_INUSE,ADJ$B_STS(R0),55$  ; Skip if slot not in use
        20 A6    91            2247  6299          CMPB    LPD$B_PTH_INX(R6),-             ; Is it on this NI?
        02 A0                  224A  6300                  ADJ$B_LPD_INX(R0)
              21    12        224C  6301          BNEQ    55$                             ; If not, skip it
              0A    ED        224E  6302          CMPZV   #TR4$V_ADDR_AREA,-              ; Is it in our area?
59    04 A0    06            2250  6303                  #TR4$S_ADDR_AREA,ADJ$W_PNA(R0),R9
              19    12        2254  6304          BNEQ    55$                             ; If not, skip it
  52    0C A0    91            2256  6305          CMPB    ADJ$B_BCPRI(R0),R2              ; Higher priority?
              13    1F        225A  6306          BLSSU   55$                             ; Branch if not
              06    1A        225C  6307          BGTRU   52$                             ; Branch if so
  53    04 A0    B1            225E  6308          CMPW    ADJ$W_PNA(R0),R3                ; If equal, compare addresses
              0B    1B        2262  6309          BLEQU   55$                             ; If address lower, skip it
  53    04 A0    3C            2264  6310 52$:     MOVZWL  ADJ$W_PNA(R0),R3                ; Update "current DRT address"
  52    0C A0    9A            2268  6311          MOVZBL  ADJ$B_BCPRI(R0),R2              ; Update "highest priority"
     58    57    D0            226C  6312          MOVL    R7,R8                           ; Update "current DRT index"
CB 57    51    F3            226F  6313 55$:     AOBLEQ  R1,R7,50$                       ; Loop thru all routers
     51    58    D0            2273  6314          MOVL    R8,R1                           ; Return DRT index in R1
        0380 8F    BA         2276  6315          POPR    #^M<R7,R8,R9>                   ; Restore registers
              05    05        227A  6316          RSB
```

```
                        227B  6318          .SBTTL  ACT_QIO_SHUT - Shutdown the datalink
                        227B  6319 ;+
                        227B  6320 ; ACT_QIO_SHUT   -  Shutdown the datalink
                        227B  6321 ;
                        227B  6322 ; INPUTS:        R11     CRI CNR ptr
                        227B  6323 ;                R10     CRI CNF ptr
                        227B  6324 ;                R6      LPD ptr
                        227B  6325 ;                R5      WQE address
                        227B  6326 ;                R4      RCB address
                        227B  6327 ;
                        227B  6328 ; OUTPUTS:       R5      Unchanged
                        227B  6329 ;                R1      Next event to be processed
                        227B  6330 ;                R0      Low bit set if state change is permitted,
                        227B  6331 ;                        Low bit clear to avoid state change
                        227B  6332 ;
                        227B  6333 ;                All other regs may be clobbered.
                        227B  6334 ;-
                        227B  6335 ACT_QIO_SHUT:                            ; Shut down the datalink
                        227B  6336                 ;
                        227B  6337                 ;   Reset LPD fields
                        227B  6338                 ;
                        227B  6339                 CLRBIT  LPD$V_INCOMING,LPD$W_STS(R6) ; Clear X.25 incoming call flag
          24 A6   94    2280  6340                 CLRB    LPD$B_XMTFLG(R6)        ; Clear Transport xmit flags
          FF 8F   90    2283  6341                 MOVB    #ADJ$C_PTY_UNK,-        ; Clear "our node type" for this circuit
          1D A6         2286  6342                         LPD$B_ETY(R6)           ; (setup later in DLL_UP)
                        2288  6343                 ;
                        2288  6344                 ;   Reset the main circuit adjacency, in the event that we got
                        2288  6345                 ;   here as a result of an initialization failure (since if we
                        2288  6346                 ;   exit run state, the main circuit adjacency is reset).
                        2288  6347                 ;
     58   20 A6   9A    2288  6348                 MOVZBL  LPD$B_PTH_INX(R6),R8    ; Get LPD index
           FE36   30    228C  6349                 BSBW    ADJ_DOWN                ; Reset main circuit adjacency
                        228F  6350                 ;
                        228F  6351                 ;   Reset substate for circuit
                        228F  6352                 ;
                0A      90    228F  6353                 MOVB    #NMA$C_LINSS_SYN,-      ; Change to "synchronizing" substate
              27 A6           2291  6354                         LPD$B_SUB_STA(R6)       ;
                02      E1    2293  6355                 BBC     #LPD$V_DLE,-            ; If BC then not going down for
           04 22 A6           2295  6356                         LPD$W_STS(R6),10$       ; "service" functions
                06      90    2298  6357                 MOVB    #NMA$C_LINSS_ASE,-      ; Init substate as "auto-service"
              27 A6           229A  6358                         LPD$B_SUB_STA(R6)       ;
          22 A6   07      E0    229C  6359 10$:             BBS     #LPD$V_X25,LPD$W_STS(R6),- ; If X.25, use DEACCESS, not SETMODE
                0E              22A0  6360                         X25_SHUTDOWN
                                22A1  6361                 ;
                                22A1  6362                 ;   Cancel any outstanding QIOs in progress on the datalink
                                22A1  6363                 ;
                0995    30      22A1  6364                 BSBW    RESET_CHAN              ; Cancel any lingering I/O
                                22A4  6365                 ;
                                22A4  6366                 ;   Issue a SETMODE to reset the datalink
                                22A4  6367                 ;
                  51    D4      22A4  6368                 CLRL    R1                      ; Indicate no need to extend P1 buffer
              0828      30      22A6  6369                 BSBW    NET$DLL_QIO_CO          ; Allocate and init WQE (co-routine)
          0000'8F       3C      22A9  6370                 MOVZWL  #IO$_SETMODE!-          ; Setup function code
                50              22AD  6371                         IO$M_SHUTDOWN,R0        ;
                        05      22AE  6372                 RSB                             ; Return to co-routine
                                22AF  6373
                                22AF  6374 ;
```

```
                     22AF  6375 ; Shutdown X.25 datalink using DEACCESS
                     22AF  6376 ;
                     22AF  6377 X25_SHUTDOWN:
                     22AF  6378         $GETFLD cri,l_use                    ; Get circuit usage
      05 50    E9    22BC  6379         BLBC    R0,X25_DEACCESS              ; Error if not found
   00    58    D1    22BF  6380         CMPL    R8,#NMA$C_CIRUS_PER          ; PVC?
         10    13    22C2  6381         BEQL    X25_PVC_SHUTDOWN             ; If so, use different shutdown
                     22C4  6382 X25_DEACCESS:
                     22C4  6383         CLRBIT  LPD$V_PVC_ACCESSED,-         ; Indicate circuit no longer ACCESSed
                     22C4  6384                 LPD$B_PVCFLG(R6)
         51    D4    22C9  6385         CLRL    R1                           ; Indicate no need for QIO buffer
      0803    30    22CB  6386         BSBW    NET$DLL_QIO_CO               ; Allocate and init WQE (co-routine)
50    0000'8F  3C    22CE  6387         MOVZWL  #IO$_DEACCESS,R0             ; Setup function code
         05    22D3  6388         RSB                                  ; Issue QIO and exit
                     22D4  6389
                     22D4  6390 ;
                     22D4  6391 ; Only issue a DEACCESS of a PVC if we are turning off the circuit.  This
                     22D4  6392 ; allows the X.25 level 2 software to retain its knowledge of the state of
                     22D4  6393 ; the circuit - so that if a "reset" is outstanding, our next reset will
                     22D4  6394 ; confirm it and keep the circuit in a consistent state.
                     22D4  6395 ;
                     22D4  6396
                     22D4  6397 X25_PVC_SHUTDOWN:
                     22D4  6398         $GETFLD cri,l_sta                    ; Get circuit state
      E0 50    E9    22E1  6399         BLBC    R0,X25_DEACCESS              ; Error if not found
   01    58    D1    22E4  6400         CMPL    R8,#NMA$C_STATE_OFF          ; Turning circuit off?
         DB    13    22E7  6401         BEQL    X25_DEACCESS                 ; If so, DEACCESS circuit
   51    16    D0    22E9  6402         MOVL    S^#CEV$C_IO_SUCC,R1          ; Signal IO_SUCC to go to next state
   50    01    D0    22EC  6403         MOVL    #1,R0                        ; Allow state change
         05    22EF  6404         RSB
```

```
                                22F0  6406          .SBTTL  ACT_QIO_STRT - Start the datalink
                                22F0  6407 ;+
                                22F0  6408 ; ACT_QIO_STRT    -  Startup the datalink
                                22F0  6409 ;
                                22F0  6410 ; INPUTS:         R11     CRI CNR ptr
                                22F0  6411 ;                 R10     CRI CNF ptr
                                22F0  6412 ;                 R6      LPD ptr
                                22F0  6413 ;                 R5      WQE address
                                22F0  6414 ;                 R4      RCB address
                                22F0  6415 ;
                                22F0  6416 ; OUTPUTS:        R5      Unchanged
                                22F0  6417 ;                 R1      Next event to be processed
                                22F0  6418 ;                 R0      Low bit set if state change is permitted,
                                22F0  6419 ;                         Low bit clear to avoid state change
                                22F0  6420 ;
                                22F0  6421 ;                 All other regs may be clobbered.
                                22F0  6422 ;-
                                22F0  6423 ACT_QIO_STRT:                            ; Startup the datalink
                                22F0  6424                 ;
                                22F0  6425                 ;   If there is still an I/O pending to the circuit, then wait
                                22F0  6426                 ;   a bit for remaining I/O to be rundown.
                                22F0  6427                 ;
                                22F0  6428                 ;   This is done because we may still have messages on the AQB
                                22F0  6429                 ;   from NETDRIVER relating to this circuit left to process.
                                22F0  6430                 ;   This is done BEFORE the suppression timer so that minor
                                22F0  6431                 ;   delays in processing the AQB messages do not force circuit
                                22F0  6432                 ;   recycle to wait a full suppression interval.
                                22F0  6433                 ;
                  1B A6   95    22F0  6434                 TSTB    LPD$B_ASTCNT(R6)        ; Any asynch activity outstanding?
                      05   12    22F3  6435                 BNEQ    18$                     ; If yes, then cannot continue
                  1C A6   95    22F5  6436                 TSTB    LPD$B_IRPCNT(R6)        ; Does NETDRIVER still have references?
                      06   13    22F8  6437                 BEQL    20$                     ; If NEQ, then wait for NETDRIVER
                                22FA  6438                                                 ; to wake us up with CRD event
                51    01   D0    22FA  6439 18$:            MOVL    #LEV$C_EXIT,R1          ; Exit state table immediately
                    00A3   31    22FD  6440                 BRW     90$                     ; Exit inhibiting state change
                                2300  6441 20$:            ;
                                2300  6442                 ;   Start the "start suppression" timer to prevent the circuit from
                                2300  6443                 ;   restarting too rapidly.  If the RECALL TIMER parameter is
                                2300  6444                 ;   specified, use that delay.  Otherwise, use a fixed timer value.
                                2300  6445                 ;
                      01   E3    2300  6446                 BBCS    #LPD$V_STRTIM,-         ; If BS then timer is already ticking
                   03 22 A6      2302  6447                         LPD$W_STS(R6),10$
                      0098   31    2305  6448                 BRW     80$
                51    20 A6   9A    2308  6449 10$:          MOVZBL  LPD$B_PTH_INX(R6),R1    ; Get LPD index
              51    51   10   78    230C  6450                 ASHL    #16,R1,R1               ; Shift into upper word
              51    011A 8F   B0    2310  6451                 MOVW    #<<WQE$C_QUAL_DLL>@8>!- ; Overlay QUAL and EVT fields
                                2315  6452                             LEV$C_STRT_TIM,R1   ;
            52    00002412'EF   9E    2315  6453                 MOVAB   STRT_TIMER_TICK,R2      ; Setup action routine address
                                231C  6454                 $GETFLD cri,T,rct               ; Get RECALL TIMER parameter
                      03 50   E8    2329  6455                 BLBS    R0,15$                  ; If not set, use default timer
                58    0A   D0    232C  6456                 MOVL    #TR$C_TIM_RESTRT,R8     ; Provide a default value
53  00    00989680 8F   58   7A    232F  6457 15$:            EMUL    R8,#10*1000*1000,#0,R3  ; Set start supression timer
                  DCC5'   30    2338  6458                 BSBW    WQE$RESET_TIM           ; Reset the timer
            54    00000000'EF   D0    233B  6459                 MOVL    NET$GL_PTR_VCB,R4       ; Recover RCB address
                                2342  6460                 ;
                                2342  6461                 ;   Check if the associated line is turned on
                                2342  6462                 ;
```

```
                02AF   30  2342  6463          BSBW    CHK_CIRC_START          ; Check if circuit can be started
               58 50   E9  2345  6464          BLBC    R0,80$                  ; If not allowed, exit
                       2348  6465          ;
                       2348  6466          ;  Increment the number of startup attempts we've made in trying
                       2348  6467          ;  to get this circuit initialized.  (This counter is used for
                       2348  6468          ;  outgoing X.25 calls to prevent too many charges from piling
                       2348  6469          ;  up.  It also can be used for statistics).
                       2348  6470          ;
            0B A6      96  2348  6471          INCB    LPD$B_STARTUPS(R6)      ; Increment number of startup attempts
                       234B  6472          ;
                       234B  6473          ;  If the state is set to SERVICE, setup for direct-access mode
                       234B  6474          ;
                       234B  6475          $GETFLD cri,l,sta               ; Get the "operater" state
             09 50     E9  2358  6476          BLBC    R0,60$                  ; If LBC then assume "OFF" state
          58    02     D1  235B  6477          CMPL    S^#NMA$C_STATE_SER,R8   ; Service state ?
                04     12  235E  6478          BNEQ    60$                     ; If NEQ then no
                       2360  6479          SETBIT  LPD$V_DLE,LPD$W_STS(R6) ; Else setup for direct-access
                       2364  6480 60$:      ;
                       2364  6481          ;  For X.25 circuits, use different startup sequence
                       2364  6482          ;
      03 22 A6    07     E1  2364  6483          BBC     #LPD$V_X25,LPD$W_STS(R6),65$ ; Branch if not X.25
                00B7   31  2369  6484          BRW     X25_STARTUP             ; Use different startup sequence
                       236C  6485          ;
                       236C  6486          ;  If LPD$_TOGGLE is set, then we must toggle the controller or
                       236C  6487          ;  if XM$_ERR_FATAL is set then we must restart the controller.
                       236C  6488          ;  This check must be made in the UCB.
                       236C  6489          ;
         52    28 A6    9A  236C  6490 65$:     MOVZBL  LPD$B_PLVEC(R6),R2      ; Get PLVEC index
         50    10 A6    D0  2370  6491          MOVL    LPD$L_UCB(R6),R0        ; Get device UCB
                17     13  2374  6492          BEQL    70$                     ; If EQL then none
                0C     E4  2376  6493          BBSC    #LPD$V_TOGGLE,-         ; Br if we must toggle the line
            0F 22 A6       2378  6494                  LPD$W_STS(R6),67$       ;
01  00000000'EF42      91  237B  6495          CMPB    PLVEC$AB_DEV[R2],-      ; DMC11 ?
                       2383  6496                  #DEVTRN$C_DEV_DMC       ;
                08     13  2383  6497          BEQL    70$                     ; If EQL yes, ignore XM$V_ERR_FATAL
                10     E1  2385  6498          BBC     #XM$V_ERR_FATAL,-       ; If BS the fatal controller error,
          03 44 A0       2387  6499                  UCB$L_DEVDEPEND(R0),70$ ; must toggle controller
             02A6   30  238A  6500 67$:     BSBW    TOGGLE_LINE             ; Toggle the controller
                       238D  6501                                          ; ...ignore errors
                       238D  6502          ;
                       238D  6503          ;  Reset the Circuit characteristics
                       238D  6504          ;
                       238D  6505 70$:     $CNFFLD cri,s,chr,R9            ; Identify characteristics buffer
      52    14 A6    3C  2394  6506          MOVZWL  LPD$W_CHAN(R6),R2       ; Get I/O channel
                51     D4  2398  6507          CLRL    R1                      ; Clear illegal I/O modifier mask
              DC63'    30  239A  6508          BSBW    NET$SET_QIOW            ; Get buffer and issue $QIOW
             06 50     E8  239D  6509          BLBS    R0,200$                 ; If LBS then okay, br to continue
          51    00     D0  23A0  6510 80$:     MOVL    #LEV$C_NO_EVT,R1        ; Nothing else to do
                50     94  23A3  6511 90$:     CLRB    R0                      ; Don't allow state change
                05     05  23A5  6512          RSB
                       23A6  6513 200$:    ;
                       23A6  6514          ;  Setup the "input packet limiter" value based on the number of
                       23A6  6515          ;  receive buffers assigned to the circuit's controller.  This is a
                       23A6  6516          ;  heuristic based on the idea that controller's requiring more
                       23A6  6517          ;  receive buffering (fast lines, satellite lines) also should be
                       23A6  6518          ;  allowed more local packet output buffering in order to prevent
                       23A6  6519          ;  severe performance degradation.
```

```
                        23A6  6520          ;
                        23A6  6521          $GETFLD cri,l,mwi                   ; Get maximum X.25 window size
                 3A 50  E8    23B3  6522     BLBS    R0,210$                     ; If set, use it as "input packet
                        23B6  6523          ;                                   ; limiter" for non-X.25 circuit
               0C00 8F  BB    23B6  6524     PUSHR   #^M<R10,R11>                ; Save regs
          5B 00000000'EF  D0    23BA  6525     MOVL    NET$GL_CNR_PLI,R11          ; Get PLI root block
                     5A  D4    23C1  6526     CLRL    R10                         ; Search from begining of list
                58 28 A6  9A    23C3  6527     MOVZBL  LPD$B_PLVEC(R6),R8          ; Search key is the PLVEC index
                        23C7  6528     $SEARCH egl,pli,l,plvec             ; Find PLI's CNF block
                   0D 50  E9    23D6  6529     BLBC    R0,205$                     ; If LBC then not found
                        23D9  6530     $GETFLD pli,l,bfn                   ; Get number of receive buffers
               0C00 8F  BA    23E6  6531 205$:  POPR    #^M<R10,R11>                ; Restore regs
                  03 50  E8    23EA  6532     BLBS    R0,210$                     ; If LBS then <pli,l,bfn> value exists
                58   04  D0    23ED  6533     MOVL    #4,R8                       ; Use 4 as the default
             1F A6   58  90    23F0  6534 210$:  MOVB    R8,LPD$B_XMT_IPL(R6)        ; Use it as input packet limiter
                        23F4  6535          ;
                        23F4  6536          ;   Issue startup QIO
                        23F4  6537          ;
                   51   D4    23F4  6538     CLRL    R1                          ; No I/O buffer needed
                 06D8   30    23F6  6539     BSBW    NET$DLL_QIO_CO              ; Allocate and init WQE (co-routine)
             57  28 A6   9A    23F9  6540     MOVZBL  LPD$B_PLVEC(R6),R7          ; Get PLVEC index
       01 00000000'EF47  91    23FD  6541     CMPB    PLVEC$AB_DEV[R7],-          ; DMC?
                        2405  6542                 #DEVTRN$C_DEV_DMC
                   05   12    2405  6543     BNEQ    240$                        ; If so,
            0034'C2  58  D0    2407  6544     MOVL    R8,WQE$C_LENGTH+P3(R2)      ; Setup DMC # buffers parameter in P3
                0000'8F  3C    240C  6545 240$:  MOVZWL  #IO$_SETMODE!-              ; Setup the default function code
                     50        2410  6546                  IO$M_STARTUP,R0        ;
                   05   05    2411  6547     RSB                                 ; Return to issue QIO
                        2412  6548
                        2412  6549 ;
                        2412  6550 ; The start supression timer has expired.  Another attempt at starting the
                        2412  6551 ; datalink should be initiated.
                        2412  6552 ;
                        2412  6553
                        2412  6554 STRT_TIMER_TICK:
             EA35   30  2412  6555          BSBW    FIND_WQE_CTX                ; Locate CNF, LPD, ADJ blocks
           07 50    E9  2415  6556          BLBC    R0,10$                      ; If LPD no longer exists, skip event
                        2418  6557          CLRBIT  LPD$V_STRTIM,LPD$W_STS(R6) ; Indicate timer no longer running
             E974   30  241C  6558          BSBW    PROC_EVT                    ; Process the event
             E966   30  241F  6559 10$:     BSBW    KILL_WQE                    ; Deallocate the WQE
                05   2422  6560          RSB                                 ; Return to caller
```

```
                        2423  6562
                        2423  6563 ;
                        2423  6564 ; Startup X.25 datalink
                        2423  6565 ;
                        2423  6566
                        2423  6567 X25_STARTUP:
                        2423  6568         $GETFLD cri,l,mwi                ; Get maximum X.25 window size
      03 50    E8       2430  6569         BLBS    R0,62$                   ; Use it, if specified
   58    04    D0       2433  6570         MOVL    #4,R8                    ; Use default value of 4
1F A6    58    90       2436  6571 62$:    MOVB    R8,LPD$B_XMT_IPL(R6)     ; Set input packet limiter
                        243A  6572         $GETFLD cri,l,use                ; Get circuit usage parameter
      0A 50    E9       2447  6573         BLBC    R0,80$                   ; If not present, defaulting error
                        244A  6574         $DISPATCH R8,<-                  ; Dispatch on X.25 circuit usage
                        244A  6575            <NMA$C_CIRUS_PER,100$>,-      ; Permanent virtual circuit
                        244A  6576            <NMA$C_CIRUS_OUT,200$>,-      ; Outgoing switched virtual circuit
                        244A  6577            <NMA$C_CIRUS_INC,300$>,-      ; Wait for incoming call
                        244A  6578         >
   51    00    D0       2454  6579 80$:    MOVL    #LEV$C_NO_EVT,R1         ; No more events
         50    D4       2457  6580         CLRL    R0                       ; Do not allow state change
               05       2459  6581         RSB
                        245A  6582
                        245A  6583 ;
                        245A  6584 ; Schedule PVC startup event.  This is done as a separate event because
                        245A  6585 ; PVC startup is a multiple step process, since 3 QIOs have to be issued
                        245A  6586 ;
      07    E0          245A  6587 100$:   BBS     #LPD$V_PVC_ACCESSED,-    ; Is circuit already ACCESSed?
   0B 25 A6              245C  6588                 LPD$B_PVCFLG(R6),150$    ; then skip this step
               88       245F  6589         BISB    #LPD$M_PVC_ACCESS!-      ; For X.25 startup, schedule access,
                        2460  6590                  LPD$M_PVC_RESTRT!-      ; restart, and
                        2460  6591                  LPD$M_PVC_RESET,-       ; reset operations for the
25 A6    07             2460  6592                 LPD$B_PVCFLG(R6)         ; next time we start the X.25 datalink
   51    18    D0       2463  6593         MOVL    #LEV$C_PVC_START,R1      ; Signal PVC startup needed
   50    01    D0       2466  6594         MOVL    #1,R0                    ; Allow state change
               05       2469  6595         RSB
                        246A  6596
   51    10    D0       246A  6597 150$:   MOVL    #LEV$C_LIN_UP,R1         ; Signal circuit is "up"
   50    01    D0       246D  6598         MOVL    #1,R0                    ; Allow state change
               05       2470  6599         RSB
                        2471  6600
                        2471  6601 ;
                        2471  6602 ; Make outgoing switched call for X.25 datalink
                        2471  6603 ;
                        2471  6604 200$:   ;
                        2471  6605         ;   If we have attempted to make the outgoing call too many times
                        2471  6606         ;   (controlled by the MAXIMUM RECALLS parameter), then give up
                        2471  6607         ;   and marked the circuit "failed".  This is done, rather than
                        2471  6608         ;   turning off the circuit and deallocating the LPD, so that
                        2471  6609         ;   the counters remain around afterwards.
                        2471  6610         ;
                        2471  6611         $GETFLD cri,l,mrc                ; Get MAXIMUM RECALLS parameter
      0F 50    E9       247E  6612         BLBC    R0,210$                  ; If not set, allow infinite retry
58    0B A6    91       2481  6613         CMPB    LPD$B_STARTUPS(R6),R8    ; Have we exceed the maximum?
         09    1B       2485  6614         BLEQU   210$                     ; If not, let it go
      0B A6    94       2487  6615         CLRB    LPD$B_STARTUPS(R6)       ; Reset # startup attempts
   51    1C    D0       248A  6616         MOVL    #LEV$C_FAILED,R1         ; Mark circuit "failed"
         50    D4       248D  6617         CLRL    R0                       ; Do not change state
               05       248F  6618         RSB                              ; Process next event
```

```
                     2490  6619          ;
                     2490  6620          ;   Call PSI to make the call
                     2490  6621          ;
                     2490  6622 210$:     $GETFLD cri,l,mbl                   ; Get MAXIMUM DATA parameter
            58   DD  249D  6623          PUSHL   R8                          ; Push value (0 if not set)
                     249F  6624          $GETFLD cri,l,mwi                   ; Get MAXIMUM WINDOW parameter
            58   DD  24AC  6625          PUSHL   R8                          ; Push value (0 if not set)
                     24AE  6626          $GETFLD cri,s,num                   ; Get remote DTE address
          52 50  E9  24BB  6627          BLBC    R0,290$                     ; If not present, defaulting error
        59   5E  D0  24BE  6628          MOVL    SP,R9                       ; Point to MWI, MBL longwords
        51   2C  D0  24C1  6629          MOVL    #8+21+15,R1                 ; Set length of extra QIO buffer
           060A  30  24C4  6630          BSBW    NET$DLL_QIO_CO              ; Allocate and init WQE (co-routine)
  0038'C2   53  D0  24C7  6631          MOVL    R3,WQE$C_LENGTH+P2(R2)      ; Set P2 to NCB descriptor
    50   08 A3  9E  24CC  6632          MOVAB   8(R3),R0                    ; Point to actual P2 buffer
        83   50  CE  24D0  6633          MNEGL   R0,(R3)+                    ; Construct NCB descriptor
        83   50  D0  24D3  6634          MOVL    R0,(R3)+
             69  D5  24D6  6635          TSTL    (R9)                        ; MAXIMUM WINDOW specified?
             09  13  24D8  6636          BEQL    230$                        ; Branch if not
        83   08  B0  24DA  6637          MOVW    #8,(R3)+                    ; Set size of item
        83   16  B0  24DD  6638          MOVW    #PSI$C_NCB_WINSIZE,(R3)+    ; Set item code
        83   69  D0  24E0  6639          MOVL    (R9),(R3)+                  ; Store maximum window size
          04 A9  D5  24E3  6640 230$:     TSTL    4(R9)                       ; MAXIMUM DATA specified?
             0A  13  24E6  6641          BEQL    240$                        ; Branch if not
        83   08  B0  24E8  6642          MOVW    #8,(R3)+                    ; Set size of item
        83   15  B0  24EB  6643          MOVW    #PSI$C_NCB_PKTSIZE,(R3)+    ; Set item code
     83   04 A9  D0  24EE  6644          MOVL    4(R9),(R3)+                 ; Store maximum packet size
  83 57   05  A1  24F2  6645 240$:     ADDW3   #5,R7,(R3)+                 ; Set size of item
        83   01  B0  24F6  6646          MOVW    #PSI$C_NCB_REMDTE,(R3)+     ; Set item code
             34  BB  24F9  6647          PUSHR   #^M<R2,R4,R5>               ; Save registers
        83   57  90  24FB  6648          MOVB    R7,(R3)+                    ; Move byte count of DTE string
  63   68   57  28  24FE  6649          MOVC    R7,(R8),(R3)                ; Move DTE address into NCB
             34  BA  2502  6650          POPR    #^M<R2,R4,R5>               ; Restore registers
  0038'D2   53  C0  2504  6651          ADDL    R3,@WQE$C_LENGTH+P2(R2)     ; Set size in P2 descriptor
  50   0000'8F  3C  2509  6652          MOVZWL  #IO$_ACCESS,R0              ; Setup I/O function code
             9E  16  250E  6653          JSB     @(SP)+                      ; Queue I/O
        5E   08  C0  2510  6654 290$:     ADDL    #8,SP                       ; Pop max window & max data
        51   00  D0  2513  6655          MOVL    #LEV$C_NO_EVT,R1            ; No more events
             05  2516  6656          RSB                                 ; Exit with R0 set true/false
                     2517  6657
                     2517  6658 ;
                     2517  6659 ; Mark the circuit as being able to accept incoming X.25 calls
                     2517  6660 ;
                     2517  6661 300$:     SETBIT  LPD$V_INCOMING,LPD$W_STS(R6) ; Mark circuit waiting for call
          FF35   31  251C  6662          BRW     80$                         ; Exit without any state change
```

```
                                251F  6664          .SBTTL  ACT_PVC_START - Start an X.25 PVC in multiple steps
                                251F  6665 ;+
                                251F  6666 ; ACT_PVC_START - Start a PVC in multiple steps
                                251F  6667 ;
                                251F  6668 ; Inputs:
                                251F  6669 ;
                                251F  6670 ;       R11 = CRI CNR address
                                251F  6671 ;       R10 = CRI CNF address
                                251F  6672 ;       R6 = LPD address
                                251F  6673 ;       R5 = WQE address
                                251F  6674 ;       R4 = RCB address
                                251F  6675 ;
                                251F  6676 ; Outputs:
                                251F  6677 ;
                                251F  6678 ;       R1 = Next event to be processed
                                251F  6679 ;       R0 = True if state change allowed, false if not.
                                251F  6680 ;-
                                251F  6681 ACT_PVC_START:
                                251F  6682 ;
                                251F  6683 ;   Determine the next step to be done in the startup process
                                251F  6684 ;
     52    25 A6    04    00    EA  251F  6685          FFS     #0,#4,LPD$B_PVCFLG(R6),R2 ; Get next thing we have to do
                                2525  6686                                            ; (NOTE: only use low nibble)
                          0A    13  2525  6687          BEQL    900$                    ; If nothing left, startup is complete
                                2527  6688          $DISPATCH R2,<-                   ; Dispatch on flag
                                2527  6689             <LPD$V_PVC_ACCESS,110$>,-      ; Issue the IO$_ACCESS function
                                2527  6690             <LPD$V_PVC_RESTRT,120$>,-      ; Issue a "restart confirmation"
                                2527  6691             <LPD$V_PVC_RESET,130$>,-       ; Issue a "reset" or "reset confirmation"
                                2527  6692          >
                    51    10    D0  2531  6693 900$:    MOVL    #LEV$C_LIN_UP,R1        ; Signal that startup is complete
                    50    01    D0  2534  6694          MOVL    #1,R0                   ; Allow state change
                          05        2537  6695          RSB
                                2538  6696 ;
                                2538  6697 ; Issue IO$_ACCESS function to access the PVC
                                2538  6698 ;
                                2538  6699 110$:    $GETFLD cri,s,nam               ; Get PVC name
                       70 50    E9  2545  6700          BLBC    R0,80$                  ; If not present, CNF is not right
                                2548  6701          CLRBIT  LPD$V_PVC_ACCESS,LPD$B_PVCFLG(R6) ; Indicate this work is done
                    51    1C    D0  254C  6702          MOVL    #8+5+T5,R1              ; Set length of extra QIO buffer
                        057F    30  254F  6703          BSBW    NET$DLL_QIO_CO          ; Allocate and init WQE (co-routine)
              0038'C2     53    D0  2552  6704          MOVL    R3,WQE$C_LENGTH+P2(R2)  ; Set P2 to NCB descriptor
              83    57    05    C1  2557  6705          ADDL3   #5,R7,(R3)+             ; Construct NCB descriptor
                 83    04    A3  9E  255B  6706          MOVAB   4(R3),(R3)+
              83    57    05    A1  255F  6707          ADDW3   #5,R7,(R3)+             ; Set size of item
                    83    18    B0  2563  6708          MOVW    #PSI$C_NCB_PVCNAM,(R3)+ ; Set item code
                    3C    BB  2566  6709          PUSHR   #^M<R2,R3,R4,R5>        ; Save registers
                    83    57    90  2568  6710          MOVB    R7,(R3)+                ; Move byte count of PVC name
              63    68    57    28  256B  6711          MOVC    R7,(R8),(R3)            ; Move PVC name into NCB
                    3C    BA  256F  6712          POPR    #^M<R2,R3,R4,R5>        ; Restore registers
              50    0000'8F   3C  2571  6713          MOVZWL  #IO$_ACCESS,R0          ; Setup I/O function code
                 10 A2    18    90  2576  6714          MOVB    #LEV$C_PVC_START,WQE$B_EVT(R2) ; Return here if I/O successful
                          05        257A  6715          RSB                             ; Issue I/O and exit
                                257B  6716 ;
                                257B  6717 ; Issue "restart confirmation" on the PVC, and ignore any error if there
                                257B  6718 ; is no restart to confirm.  This is done because the PVC is always active,
                                257B  6719 ; and a restart operation puts the circuit into a known state.
                                257B  6720 ;
```

```
                     257B  6721 120$:   SETBIT  LPD$V_PVC_ACCESSED,-    ; Indicate PVC now ACCESSed
                     257B  6722                 LPD$B_PVCFLG(R6)
                     2580  6723           CLRBIT  LPD$V_PVC_RESTRT,LPD$B_PVCFLG(R6) ; Indicate this work is done
           51    D4  2584  6724           CLRL    R1                      ; No QIO buffer needed
         0548    30  2586  6725           BSBW    NET$DLL_QIO_CO          ; Allocate and init WQE (co-routine)
0030'C2    07    D0  2589  6726           MOVL    #PSI$C_RESTART,WQE$C_LENGTH+P4(R2) ; Set P4 to "restart"
50    0000'8F    3C  258E  6727           MOVZWL  #IO$_NETCONTROL,R0      ; Set I/O function code
  10 A2    18    90  2593  6728           MOVB    #LEV$C_PVC_START,WQE$B_EVT(R2) ; Return here if I/O successful
  14 A2    18    90  2597  6729           MOVB    #LEV$C_PVC_START,WQE$L_PM2(R2) ; Return here if I/O fails too
           05    05  259B  6730           RSB                             ; Issue I/O and exit
                     259C  6731 ;
                     259C  6732 ; Issue "reset" or "reset confirmation" on the PVC, and ignore any errors.
                     259C  6733 ; This is done to clear any outstanding received messages from the previous
                     259C  6734 ; user of the PVC.
                     259C  6735 ;
                     259C  6736 130$:   CLRBIT  LPD$V_PVC_RESET,LPD$B_PVCFLG(R6) ; Indicate this work is done
           51    D4  25A0  6737           CLRL    R1                      ; No QIO buffer needed
         052C    30  25A2  6738           BSBW    NET$DLL_QIO_CO          ; Allocate and init WQE (co-routine)
0030'C2    03    D0  25A5  6739           MOVL    #PSI$C_RESET,WQE$C_LENGTH+P4(R2) ; Set P4 to "reset"
50    0000'8F    3C  25AA  6740           MOVZWL  #IO$_NETCONTROL,R0      ; Set I/O function code
  10 A2    18    90  25AF  6741           MOVB    #LEV$C_PVC_START,WQE$B_EVT(R2) ; Return here if I/O successful
  14 A2    18    90  25B3  6742           MOVB    #LEV$C_PVC_START,WQE$L_PM2(R2) ; Return here if I/O fails too
           05    05  25B7  6743           RSB                             ; Issue I/O and exit
                     25B8  6744
                     25B8  6745 ;
                     25B8  6746 ; Come here if an error was encountered before beginning the startup
                     25B8  6747 ; to abort the operation, and wait until the supression timer causes it to
                     25B8  6748 ; be tried again.
                     25B8  6749 ;
           51    00    D0  25B8  6750 80$:    MOVL    #LEV$C_NO_EVT,R1        ; No more events
                 50    D4  25BB  6751           CLRL    R0                      ; Don't allow state change
                       05  25BD  6752           RSB
```

```
                                25BE  6754          .SBTTL  ACT_X25_CALL - Accept incoming X.25 call
                                25BE  6755 ;+
                                25BE  6756 ; ACT_X25_CALL - Accept incoming X.25 call
                                25BE  6757 ;
                                25BE  6758 ; This circuit has already been determined to be waiting for an incoming
                                25BE  6759 ; call and allocated for that purpose.  All we have to do for this event
                                25BE  6760 ; is issue the ACCEPT QIO.  On successful I/O completion, the transition
                                25BE  6761 ; will be made to the Routing Initialization state.
                                25BE  6762 ;
                                25BE  6763 ; Inputs:
                                25BE  6764 ;
                                25BE  6765 ;       R11 = CRI CNR address
                                25BE  6766 ;       R10 = CRI CNF address
                                25BE  6767 ;       R6 = LPD address
                                25BE  6768 ;       R5 = WQE address
                                25BE  6769 ;       R4 = RCB address
                                25BE  6770 ;
                                25BE  6771 ; Outputs:
                                25BE  6772 ;
                                25BE  6773 ;       R1 = Next event to be processed
                                25BE  6774 ;       R0 = True if state change allowed, false if not.
                                25BE  6775 ;-
                                25BE  6776 ACT_X25_CALL:
      51    14 A5    3C        25BE  6777          MOVZWL  WQE$L_PM2(R5),R1        ; Get size of X.25 NCB
         51    08    C0        25C2  6778          ADDL    #8,R1                   ; Add in size of NCB descriptor
            0509    30        25C5  6779          BSBW    NET$DLL_QIO_CO          ; Allocate and init WQE (co-routine)
 0038'C2     53    D0        25C8  6780          MOVL    R3,WQE$C_LENGTH+P2(R2)  ; Set P2 to NCB descriptor
   83    14 A5    3C        25CD  6781          MOVZWL  WQE$L_PM2(R5),(R3)+     ; Construct NCB descriptor
   83    04 A3    9E        25D1  6782          MOVAB   4(R3),(R3)+
         3C       BB        25D5  6783          PUSHR   #^M<R2,R3,R4,R5>        ; Save registers
      14 A5       28        25D7  6784          MOVC    WQE$L_PM2(R5),-         ; Move NCB into I/O WQE
   63    24 A5               25DA  6785                  WQE$C_LENGTH(R5),(R3)
         3C       BA        25DD  6786          POPR    #^M<R2,R3,R4,R5>        ; Restore registers
         13       10        25DF  6787          BSBB    CHK_CIRC_START          ; Check if circuit can be started
      06 50       E9        25E1  6788          BLBC    R0,TO$                  ; Branch if not
50    0000'8F     3C        25E4  6789          MOVZWL  #IO$_ACCESS!IO$M_ACCEPT,R0 ; Setup I/O function code
                  05        25E9  6790          RSB                             ; Issue QIO and exit
                            25EA  6791 ;
                            25EA  6792 ; Circuit is not in a state to be started - change QIO to be a "REJECT"
                            25EA  6793 ; and on I/O completion, cause the circuit to be recycled.
                            25EA  6794 ;
50    0000'8F     3C        25EA  6795 IO$:     MOVZWL  #IO$_ACCESS!IO$M_ABORT,R0 ; Setup I/O function code
   10 A2    04    90        25EF  6796          MOVB    #LEV$C_REQ_SHUT,WQE$B_EVT(R2) ; Recycle circuit on success
                  05        25F3  6797          RSB                             ; Issue QIO and exit
```

```
                                        .SBTTL  CHK_CIRC_START - Check if circuit can be started
                                ;+
                                ; CHK_CIRC_START - Check if circuit can be started
                                ;
                                ; Determine if the circuit can be started by examining the associated
                                ; line state, and by making sure that the ACP's state is on.
                                ;
                                ; Inputs:
                                ;
                                ;       R4 = RCB address
                                ;       R6 = LPD address
                                ;
                                ; Outputs:
                                ;
                                ;       R0 = True if startup allowed, else false
                                ;
                                ;       No other registers are destroyed.
                                ;-
                                CHK_CIRC_START:
                     52   DD    25F4  6818              PUSHL   R2                      ; Save registers
                                25F6  6819              ;
                                25F6  6820              ;   Check to make sure that the associated line is "on"
                                25F6  6821              ;
                   52   28 A6   9A  25F6  6822              MOVZBL  LPD$B_PLVEC(R6),R2      ; Get the associated line index
                 00000000'EF    16  25FA  6823              JSB     NET$GET_VEC2            ; Setup the line
                        2A 50   E9  2600  6824              BLBC    R0,80$                  ; If LBC then setup failed
              0A 22 A6    07    E0  2603  6825              BBS     #LPD$V_X25,LPD$W_STS(R6),25$ ; If X.25, there is no assoc. line
          00  00000000'EF42    91  2608  6826              CMPB    PLVEC$AB_STATE[R2],-    ; Is the line "on"
                                2610  6827                      #NMA$C_STATE_ON         ;
                     1B   12    2610  6828              BNEQ    80$                     ; If NEQ no, can't start circuit
                                2612  6829              ;
                                2612  6830              ;   Check to make sure that the ACP state is not "off" or "init".
                                2612  6831              ;
                                2612  6832  25$:        $DISPATCH TYPE=B,RCB$B_STI(R4),<-
                                2612  6833                      <ACP$C_STA_I, 80$>,-    ; Initializing
                                2612  6834                      <ACP$C_STA_N, 50$>,-    ; On
                                2612  6835                      <ACP$C_STA_R, 50$>,-    ; Restricted
                                2612  6836                      <ACP$C_STA_S, 30$>,-    ; Shut
                                2612  6837                      <ACP$C_STA_F, 80$>,-    ; Off
                                2612  6838                      <ACP$C_STA_H, 80$>,-    ; Hibernating (due to bug)
                                2612  6839              >
                   54 A4    B5  2623  6840  30$:        TSTW    RCB$W_MCOUNT(R4)        ; Time to shut down ?
                      05    13  2626  6841              BEQL    80$                     ; If so, go away
          50          01    D0  2628  6842  50$:        MOVL    #1,R0                   ; Successful
                      02    11  262B  6843              BRB     90$
                                262D  6844
                      50    D4  262D  6845  80$:        CLRL    R0                      ; Do not allow circuit on
                   52 8ED0    262F  6846  90$:        POPL    R2                      ; Restore registers
                         05    2632  6847              RSB
```

```
                                  2633          .SBTTL  TOGGLE_LINE - Shutdown and startup line
                                  2633  ;+
                                  2633  ; TOGGLE_LINE - Toggle line state
                                  2633  ;
                                  2633  ; The line is indicating a fatal device error.  Turn the device
                                  2633  ; OFF and ON in order to re-initialize it.
                                  2633  ;
                                  2633  ; Inputs:
                                  2633  ;
                                  2633  ;       R2 = PLVEC index
                                  2633  ;
                                  2633  ; Outputs:
                                  2633  ;
                                  2633  ;       R0 = Status from startup operation
                                  2633  ;-
                                  2633  TOGGLE_LINE:                                  ; Toggle line state
              0C64 8F     BB      2633          PUSHR   #^M<R2,R5,R6,R10,R11>         ; Save crucial regs
                 56  52   D0      2637          MOVL    R2,R6                         ; Move PLVEC pointer to safe reg
     5B   00000000'EF     D0      263A          MOVL    NET$GL_CNR_PLI,R11            ; Get PLI root block
                    5A   D4      2641          CLRL    R10                           ; Search from begining of list
                 58  56   D0      2643          MOVL    R6,R8                         ; Search key is the PLVEC index
                                  2646          $SEARCH eql,pli,l,plvec               ; Find PLI's CNF block
                 1C 50   E9      2655          BLBC    R0,100$                       ; If LBC then not found
     7E   00000000'EF46   9A      2658          MOVZBL  PLVEC$AB_STATE[R6],-(SP)      ; Save previous state
                    01   90      2660          MOVB    #NMA$C_STATE_OFF,-            ; Prepare to turn line off
          00000000'EF46           2662                  PLVEC$AB_STATE[R6]            ;
                    0F   10      2668          BSBB    110$                          ; Turn the line off
                                  266A                                                ; ...ignore errors
                    8E   F6      266A          CVTLB   (SP)+,-                       ; Prepare to turn line back on
          00000000'EF46           266C                  PLVEC$AB_STATE[R6]            ;
                    05   10      2672          BSBB    110$                          ; Turn the line on
              0C64 8F     BA      2674  100$:   POPR    #^M<R2,R5,R6,R10,R11>         ; Restore regs
                        05      2678          RSB                                   ; Return status in R0
                                  2679
     52   00000000'EF46   3C      2679  110$:   MOVZWL  PLVEC$AW_CHAN[R6],R2          ; Get I/O channel
                    51   D4      2681          CLRL    R1                            ; Clear "illegal" I/O fct code mask
                                  2683          $CNFFLD pli,s,chr,R9                  ; Setup characteristics buffer i.d.
                 D973'   30      268A          BSBW    NET$SET_QIOW                  ; Turn the line on
                        05      268D          RSB
```

```
                                268E  6889          .SBTTL  ACT_XMT - Transmit pending messages
                                268E  6890 ;+
                                268E  6891 ; ACT_XMT   -   Conditionally xmit a message
                                268E  6892 ;
                                268E  6893 ; INPUTS:       R11     CRI CNR ptr
                                268E  6894 ;               R10     CRI CNF ptr
                                268E  6895 ;               R7      ADJ address
                                268E  6896 ;               R6      LPD address
                                268E  6897 ;               R5      WQE address
                                268E  6898 ;               R4      RCB address
                                268E  6899 ;
                                268E  6900 ; OUTPUTS:      R5      Unchanged
                                268E  6901 ;               R1      Next event to be processed
                                268E  6902 ;               R0      Low bit set if state change is permitted,
                                268E  6903 ;                       Low bit clear to avoid state change
                                268E  6904 ;
                                268E  6905 ;               All other regs may be clobbered.
                                268E  6906 ;-
                                268E  6907 ACT_XMT:                                     ; Xmit message if possible
                          0422  30    268E  6908          BSBW    CHK_IO                ; Okay to xmit?
                        1A 50   E9    2691  6909          BLBC    R0,10$                ; If LBC no
52    24 A6     08      00      EA    2694  6910          FFS     #0,#8,LPD$B_XMTFLG(R6),R2 ; Get xmit flag
                          12    13    269A  6911          BEQL    10$                   ; If EQL then none set
                                269C  6912          $DISPATCH R2,<-                       ; Dispatch on xmit flag
                                269C  6913              <LPD$V_XMT_DALLY,XMT_DALLY>,- ; Wait to send start message
                                269C  6914              <LPD$V_XMT_STR,  XMT_STR>,- ; Transport "Start" message
                                269C  6915              <LPD$V_XMT_VRF,  XMT_VRF>,- ; Transport "Verification" message
                                269C  6916              <LPD$V_XMT_RT,   XMT_RT>,-  ; Transport "Routing" message
                                269C  6917              <LPD$V_XMT_ART,  XMT_ART>,- ; Transport "Area Routing" message
                                269C  6918              <LPD$V_XMT_IDLE, 100$>,-    ; All Transport init messages xmitted
                                269C  6919          >
51    01        D0    26AE  6920 10$:    MOVL    #LEV$C_EXIT,R1                ; Nothing to do, exit state table
50    01        D0    26B1  6921         MOVL    #1,R0                         ; Allow state change
                    05  26B4  6922         RSB                                   ;
                        26B5  6923
                        26B5  6924 100$:   CLRBIT  LPD$V_XMT_IDLE,-              ; Clear the flag
                        26B5  6925                 LPD$B_XMTFLG(R6)              ;
51    0F        D0    26B9  6926         MOVL    S^#LEV$C_XMT_IDLE,R1          ; Xmitter is idle during Transport init
50    01        D0    26BC  6927         MOVL    #1,R0                         ; Allow state change
                    05  26BF  6928         RSB                                   ;
```

```
                                26C0  6930          .SBTTL  XMT_DALLY - Dally before sending start message
                                26C0  6931 ;+
                                26C0  6932 ; XMT_DALLY - Dally before sending start message
                                26C0  6933 ;
                                26C0  6934 ; This routine is called to dally for a while before sending out the
                                26C0  6935 ; Start message.  This is so that we can properly initialize with older
                                26C0  6936 ; nodes which do not properly parse/ignore Phase IV start messages.  By
                                26C0  6937 ; dallying a while before sending, it gives us a chance to hear his start
                                26C0  6938 ; message, and send the correct version of the message based on the type
                                26C0  6939 ; of message he sends.
                                26C0  6940 ;
                                26C0  6941 ; Inputs:
                                26C0  6942 ;
                                26C0  6943 ;       R11 = CRI CNR address
                                26C0  6944 ;       R10 = CRI CNF address
                                26C0  6945 ;       R7 = ADJ address
                                26C0  6946 ;       R6 = LPD address
                                26C0  6947 ;
                                26C0  6948 ; Outputs:
                                26C0  6949 ;
                                26C0  6950 ;       None
                                26C0  6951 ;-
                                26C0  6952 XMT_DALLY:
          51    20 A6     3C    26C0  6953          MOVZWL  LPD$W_PTH(R6),R1        ; Get LPD ID
        51    51    10    78    26C4  6954          ASHL    #16,R1,R1               ; Shift into upper word (REQIDT)
        51    0100 8F     B0    26C8  6955          MOVW    #<WQE$C_QUAL_DLL@8>!-   ; Setup timer qualifier
                                26CD  6956                  LEV$C_NO_EVT,R1         ; and timer event
53  00000000 01312D00 8F     7D    26CD  6957          MOVQ    #TR$C_TIM_DALLY*-       ; Set dally timer
                                26D8  6958                  10*1000*1000,R3
        52    E6A4 CF     9E    26D8  6959          MOVAB   NET$DLL_PRC_WQE,R2      ; and process event when it fires
                D920'     30    26DD  6960          BSBW    WQE$RESET_TIM           ; Start timer
                                26E0  6961          CLRBIT  #LPD$V_XMT_DALLY,-      ; We've done this now
                                26E0  6962                  LPD$B_XMTFLG(R6)
        51    01    D0    26E5  6963          MOVL    #LEV$C_EXIT,R1          ; Exit state table immediately
        50    01    D0    26E8  6964          MOVL    #1,R0                   ; Allow state change (if any)
                    05    26EB  6965          RSB
```

```
                                26EC  6967          .SBTTL  XMT_STR - Transmit start message
                                26EC  6968 ;+
                                26EC  6969 ; XMT_STR -      Build and transmit a Transport "start" message
                                26EC  6970 ;
                                26EC  6971 ; INPUTS:        R11     CRI CNR ptr
                                26EC  6972 ;                R10     CRI CNF ptr
                                26EC  6973 ;                R7      ADJ address
                                26EC  6974 ;                R6      LPD address
                                26EC  6975 ;                R4      RCB address
                                26EC  6976 ;
                                26EC  6977 ; OUTPUTS:       R5      Unchanged
                                26EC  6978 ;                R1      Next event to be processed
                                26EC  6979 ;                R0      Low bit set if state change is permitted,
                                26EC  6980 ;                        Low bit clear to avoid state change
                                26EC  6981 ;
                                26EC  6982 ;                All other registers may be clobbered
                                26EC  6983 ;-
                                26EC  6984 XMT_STR:                                     ; Xmt a transport initialization msg
                                26EC  6985          ASSUME  TR3C_STR_LNG  LE TR2C_STR_MXL
                                26EC  6986
          51    50 8F    9A     26EC  6987          MOVZBL  #TR2C_STR_MXL,R1            ; Setup size of P1 buffer
                  03DE   30     26F0  6988          BSBW    NET$DLL_QIO_CO              ; Call co-routine to init WQE
        003C'C2     53   D0     26F3  6989          MOVL    R3,WQE$C_LENGTH+P1(R2)      ; Point to buffer
        0038'C2     53   CE     26F8  6990          MNEGL   R3,WQE$C_LENGTH+P2(R2)      ; Bias I/O buffer size
     5B   00000000'EF   D0     26FD  6991          MOVL    NET$GL_CNR_LNI,R11          ; Set CNR for local data base
     5A   00000000'EF   D0     2704  6992          MOVL    NET$GL_PTR_LNI,R10          ; Get LNI CNF
          50    1D A6   9A     270B  6993          MOVZBL  LPD$B_ETY(R6),R0            ; Get our (adapted) "node type"
     50   0000014A'EF40   9A     270F  6994          MOVZBL  PTY_TO_PHASE[R0],R0         ; Get our (adapted) "phase"
                                2717  6995          $DISPATCH R0,<=
                                2717  6996                  <2,STR2>,-                ; Phase II
                                2717  6997                  <3,STR3>,-                ; Phase III
                                2717  6998                  <4,STR4>>                 ; Phase IV
                51    01   D0     2721  6999          MOVL    S^#LEV$C_EXIT,R1            ; Don't do anything
                      50   94     2724  7000          CLRB    R0                          ; Inhibit state change
                           05     2726  7001          RSB                                 ; Return with LBC in R0
                                2727  7002          ;
                                2727  7003          ;   Build and transmit Phase II "init" message
                                2727  7004          ;
                           57   DD     2727  7005 STR2:   PUSHL   R7                      ; Save registers
              83    58 8F    90     2729  7006          MOVB    #TR2C_MSG_INI,(R3)+         ; Enter message type code
                    83   01    90     272D  7007          MOVB    #TR2C_INI_STR,(R3)+         ; Enter message sub-type code
                                2730  7008          ;
                           00   EF     2730  7009          EXTZV   #TR4$V_ADDR_DEST,-          ; Get our address (without area)
         50    0E A4    0A            2732  7010                  #TR4$S_ADDR_DEST,RCB$W_ADDR(R4),R0
                    50   02    A4     2736  7011          MULW    #2,R0                       ; Start converting address to EX-2 field
                    50   02    86     2739  7012          DIVB2   #2,R0                       ; Now bits 7-15 are shifted
     83     50    0080 8F    A9     273C  7013          BISW3   #128,R0,(R3)+               ; Set the extend bit and enter it
                                2742  7014          $CNFFLD lni,s,nam,R9                ; Identify local node name field
                      00A1  30     2749  7015          BSBW    MOVIT                       ; Fetch and enter the string
                    83   00    90     274C  7016          MOVB    #TR2C_STR_FCT,(R3)+         ; Enter supported functions
                    83   07    90     274F  7017          MOVB    #TR2C_STR_REQ!-             ;
                                2752  7018                    TR2M_REQ_VRF,(R3)+        ; Enter 'requests'
                  83    50 A6   B0     2752  7019          MOVW    LPD$W_BUFSIZ(R6),(R3)+      ; Enter block size
                  83    7C A4   B0     2756  7020          MOVW    RCB$W_ECLSEGSIZ(R4),(R3)+   ; Enter NSP segment size
                  83    58 A4   B0     275A  7021          MOVW    RCB$W_MAX_LNK(R4),(R3)+     ; Enter max links
               83   0103 8F    B0     275E  7022          MOVW    #^X<0103>,(R3)+             ; Enter PhaseII compatable routing
                           83   94     2763  7023          CLRB    (R3)+                       ; version (3.1.0)
```

```
                83    0103 8F    B0   2765  7024          MOVW    #^X<0103>,(R3)+         ; Enter PhaseII compatable NSP version
                      83   94    B0   276A  7025          CLRB    (R3)+                   ; (3.1.0)
                                 276C  7026          $CNFFLD lni,s,ide,R9            ; Identify system version field
                      78   10    2773  7027          BSBB    MOVIT                   ; Fetch and enter the string
                      57 8ED0    2775  7028          POPL    R7                      ; Restore registers
                      66   11    2778  7029          BRB     XMT                     ; Return to co-routine to xmt the msg
                                 277A  7030          ;
                                 277A  7031          ;   Build Phase III "start" message
                                 277A  7032          ;
                      83   01    90   277A  7033 STR3:  MOVB    #TR3C_MSG_STR,(R3)+     ; Enter msg type code
                      00   EF    277D  7034          EXTZV   #TR4$V_ADDR_DEST,-      ; Get our address (without area)
                50  0E A4   0A       277F  7035                  #TR4$S_ADDR_DEST,RCB$W_ADDR(R4),R0
                      83   50    B0   2783  7036          MOVW    R0,(R3)+                ; Enter the local address in msg
                                 2786  7037          $DISPATCH LPD$B_ETY(R6),TYPE=B,<-
                                 2786  7038                  <ADJ$C_PTY_PH3,2$>,-    ; Phase III routing
                                 2786  7039                  <ADJ$C_PTY_PH3N,3$>>    ; Phase III endnode
                      83   02    90   278F  7040 2$:     MOVB    #TR3C_NTY_PH3,(R3)+     ; Set local node type (Level 1 Routing)
                      03   11    2792  7041          BRB     10$
                      83   03    90   2794  7042 3$:     MOVB    #TR3C_NTY_PH3N,(R3)+    ; Set local node type (Nonrouting)
                                 2797  7043 10$:    SETBIT  #TR3V_REQ_VRF,-1(R3)    ; request for verification
                   83   50 A6   B0   279C  7044          MOVW    LPD$W_BUFSIZ(R6),(R3)+  ; Enter block size
                83 0301 8F    B0   27A0  7045          MOVW    #TR3C_TIVER,(R3)+       ; Enter routing version number
                      83   00    90   27A5  7046          MOVB    #0,(R3)+                ; Enter user ECO number
                      83   94    27A8  7047          CLRB    (R3)+                   ; Enter the verification seed
                      34   11    27AA  7048          BRB     XMT                     ; Transmit the message
                                 27AC  7049          ;
                                 27AC  7050          ;   Build and transmit Phase IV "start" message
                                 27AC  7051          ;
                      83   01    90   27AC  7052 STR4:  MOVB    #TR4C_MSG_STR,(R3)+     ; Enter msg type code
                   83   0E A4    B0   27AF  7053          MOVW    RCB$W_ADDR(R4),(R3)+    ; Enter the local address
                                 27B3  7054          $DISPATCH LPD$B_ETY(R6),TYPE=B,<-
                                 27B3  7055                  <ADJ$C_PTY_AREA,1$>,-   ; Phase IV level 2 routing
                                 27B3  7056                  <ADJ$C_PTY_PH4,2$>,-    ; Phase IV routing
                                 27B3  7057                  <ADJ$C_PTY_PH4N,3$>>    ; Phase IV endnode
                      83   02    90   27BE  7058 2$:     MOVB    #TR4C_NTY_ROU,(R3)+     ; Set local node type (Level 1 Routing)
                      08   11    27C1  7059          BRB     10$
                      83   01    90   27C3  7060 1$:     MOVB    #TR4C_NTY_ARO,(R3)+     ; Set local node type (Level 2 Routing)
                      03   11    27C6  7061          BRB     10$
                      83   03    90   27C8  7062 3$:     MOVB    #TR4C_NTY_NROU,(R3)+    ; Set local node type (Nonrouting)
                                 27CB  7063 10$:    SETBIT  #TR4V_REQ_VRF,-1(R3)    ; Set request for verification
                   83   50 A6   B0   27D0  7064          MOVW    LPD$W_BUFSIZ(R6),(R3)+  ; Enter block size
                      83   02    B0   27D4  7065          MOVW    #TR4C_TIVER,(R3)+       ; Enter routing version number
                      83   00    90   27D7  7066          MOVB    #0,(R3)+                ; Enter user ECO number
                   83   18 A6   B0   27DA  7067          MOVW    LPD$W_INT_TLK(R6),(R3)+ ; Enter hello timer
                      83   94    27DE  7068          CLRB    (R3)+                   ; No optional data
                                 27E0  7069          ;
                                 27E0  7070          ;   Transmit message
                                 27E0  7071          ;
0038'C2    53   C0   27E0  7072 XMT:     ADDL    R3,WQE$C_LENGTH+P2(R2)  ; Calculate I/O buffer size
                50   00'   D0   27E5  7073          MOVL    S^#IO$_WRITELBLK,R0     ; Setup I/O function
                                 27E8  7074          CLRBIT  LPD$V_XMT_STR,-         ; No further need to send message
                                 27E8  7075                   LPD$B_XMTFLG(R6)       ;
                      05   27EC  7076          RSB                             ; Return to co-routine, then to caller
                                 27ED  7077
                                 27ED  7078
                   D810'   30   27ED  7079 MOVIT:  BSBW    CNF$GET_FIELD           ; Fetch the string
                83   57   90   27F0  7080          MOVB    R7,(R3)+                ; Enter count field
```

```
        03   11  27F3  7081          BRB     MOVITU                  ; Go to end of loop
  83    88   90  27F5  7082 MC1:     MOVB    (R8)+,(R3)+             ; Enter text without clobbering  R0-R5
     FA 57   F4  27F8  7083 MOVITU:  SOBGEQ  R7,MC1                  ; Loop for each character
             05  27FB  7084          RSB
```

```
                        27FC  7086          .SBTTL  XMT_VRF - Transmit verification message
                        27FC  7087 ;+
                        27FC  7088 ; XMT_VRF   - Build and transmit Transport verification message
                        27FC  7089 ;
                        27FC  7090 ; INPUTS:       R11     CRI CNR ptr
                        27FC  7091 ;               R10     CRI CNF ptr
                        27FC  7092 ;               R7      ADJ ptr
                        27FC  7093 ;               R6      LPD ptr
                        27FC  7094 ;               R4      RCB ptr
                        27FC  7095 ;
                        27FC  7096 ; OUTPUTS:      R5      Unchanged
                        27FC  7097 ;               R1      Next "event longword" to be processed
                        27FC  7098 ;               R0      Low bit set if state change is permitted,
                        27FC  7099 ;                       Low bit clear to avoid state change
                        27FC  7100 ;
                        27FC  7101 ;               All other registers may be clobbered
                        27FC  7102 ;-
                        27FC  7103 XMT_VRF:                                     ; Xmt Transport verification message
5B  00000000'EF   D0    27FC  7104          MOVL    NET$GL_CNR_NDI,R11          ; Set CNR for remote node data base
    58    04 A7   3C    2803  7105          MOVZWL  ADJ$W_PNA(R7),R8            ; Get partner's node address
          06      12    2807  7106          BNEQ    20$                         ; Not yet known  if EQL
       51 01      D0    2809  7107          MOVL    S^#LEV$C_EXIT,R1            ; Partner is not yet known, - i.e.,
                        280C  7108                                              ;  no 'start' message yet
          50      94    280C  7109          CLRB    R0                          ; Inhibit state change
          05      05    280E  7110          RSB                                 ; Return with LBC in R0
                        280F  7111 20$:     ;
                        280F  7112          ;   Get the transmit verification password
                        280F  7113          ;
             57   DD    280F  7114          PUSHL   R7                          ; Save ADJ address
           D7EC'  30    2811  7115          BSBW    NET$NDI_BY_ADD              ; Find NDI CNF for partner node
             57   D4    2814  7116          CLRL    R7                          ; Zero password string size assuming
                        2816  7117                                              ; no NDI was found
          0D 50   E9    2816  7118          BLBC    R0,30$                      ; If LBC then no NDI was found
                        2819  7119          $GETFLD ndi,s,tpa                   ; Get transmit password descriptor
                        2826  7120                                              ; R7,R8 = 0 on return if field is null
       58   57   7D    2826  7121 30$:     MOVQ    R7,R8                       ; Pass password descriptor in R8/R9
            57 8ED0    2829  7122          POPL    R7                          ; Restore ADJ address
                        282C  7123          ;
                        282C  7124          ;   Build and transmit the message
                        282C  7125          ;
         51   46 8F    9A    282C  7126          MOVZBL  #TR_C_VRF_LNG+2,R1          ; Setup size of I/O buffer   ;! +2 is tmp
              029E      30    2830  7127          BSBW    NET$DLL_QIO_CO              ; Call co-routine to init WQE
      003C'C2   53      D0    2833  7128          MOVL    R3,WQE$C_LENGTH+P1(R2)      ; Point to buffer
      0038'C2   53      CE    2838  7129          MNEGL   R3,WQE$C_LENGTH+P2(R2)      ; Bias I/O buffer size
        50   1D A6     9A    283D  7130          MOVZBL  LPD$B_ETY(R6),R0            ; Get our (adapted) "node type"
50  0000014A'EF40     9A    2841  7131          MOVZBL  PTY_TO_PHASE[R0],R0         ; Get our (adapted) "phase"
                        2849  7132          $DISPATCH R0,<=
                        2849  7133                  <2,60$>,-                   ; Phase II
                        2849  7134                  <3,50$>,-                   ; Phase III
                        2849  7135                  <4,40$>>                    ; Phase IV
          51   01    D0    2853  7136          MOVL    S^#LEV$C_EXIT,R1            ; Don't do anything
               50    94    2856  7137          CLRB    R0                          ; Inhibit state change
                     05    2858  7138          RSB                                 ; Return with LBC in R0
                        2859  7139          ;
                        2859  7140          ;   Build Phase IV header
                        2859  7141          ;
          83   03    90    2859  7142 40$:     MOVB    #TR4C_MSG_VRF,(R3)+         ; Enter message type code
```

```
                83   0E A4   B0  285C  7143          MOVW    RCB$W_ADDR(R4),(R3)+    ; Enter local node address
                   83   58   90  2860  7144          MOVB    R8,(R3)+                ; Enter length of password
                   50   58   D0  2863  7145          MOVL    R8,R0                   ; Setup msg psw field size
                        1E   11  2866  7146          BRB     70$                     ; Continue in common
                                 2868  7147          ;
                                 2868  7148          ;   Build Phase III header
                                 2868  7149          ;
                   83   03   90  2868  7150 50$:     MOVB    #TR3C_MSG_VRF,(R3)+     ; Enter message type code
                        00   EF  286B  7151          EXTZV   #TR4$V_ADDR_DEST,-      ; Get our address (without area)
                50   0E A4   0A  286D  7152                  #TR4$S_ADDR_DEST,RCB$W_ADDR(R4),R0
                   83   50   B0  2871  7153          MOVW    R0,(R3)+                ; Enter local node address
                   83   58   90  2874  7154          MOVB    R8,(R3)+                ; Enter length of password
                   50   58   D0  2877  7155          MOVL    R8,R0                   ; Setup msg psw field size
                        0A   11  287A  7156          BRB     70$                     ; Continue in common
                                 287C  7157          ;
                                 287C  7158          ;   Build Phase II header
                                 287C  7159          ;
                83   58 8F   90  287C  7160 60$:     MOVB    #TR2C_MSG_INI,(R3)+     ; Enter message type code
                   83   02   90  2880  7161          MOVB    #TR2C_INI_VRF,(R3)+     ; Enter message sub-type code
                   50   08   D0  2883  7162          MOVL    #8,R0                   ; Setup msg psw field size
                                 2886  7163          ;
                                 2886  7164          ;   Move the password
                                 2886  7165          ;
                        34   BB  2886  7166 70$:     PUSHR   #^M<R2,R4,R5>           ; Save regs
          63   50   00   69   58  2C  2888  7167          MOVC5   R8,(R9),#0,R0,(R3)      ; Move the password - null fill
                        34   BA  288E  7168          POPR    #^M<R2,R4,R5>           ; Restore regs
                0038'C2   53   C0  2890  7169          ADDL    R3,WQE$C_LENGTH+P2(R2)  ; Calculate I/O buffer size
                   50   00'  D0  2895  7170          MOVL    S^#IO$_WRITELBLK,R0     ; Setup I/O function
                                 2898  7171          CLRBIT  LPD$V_XMT_VRF,-         ; No further need to send message
                                 2898  7172                   LPD$B_XMTFLG(R6)       ;
                        05   289C  7173          RSB                             ; Return to co-routine, then to caller
```

```
                                289D  7175          .SBTTL  XMT_RT - Transmit a routing message
                                289D  7176 ;+
                                289D  7177 ; XMT_RT - Transmit a routing message
                                289D  7178 ;
                                289D  7179 ;   INPUTS:      R11     CRI CNR ptr
                                289D  7180 ;                R10     CRI CNF ptr
                                289D  7181 ;                R7      ADJ ptr
                                289D  7182 ;                R6      LPD ptr
                                289D  7183 ;                R4      RCB address
                                289D  7184 ;
                                289D  7185 ;   OUTPUTS:     R5      Unchanged
                                289D  7186 ;                R1      Next event to be processed
                                289D  7187 ;                R0      Low bit set if state change is permitted,
                                289D  7188 ;                        Low bit clear to avoid state change
                                289D  7189 ;
                                289D  7190 ;                All other regs may be clobbered
                                289D  7191 ;-
                                289D  7192 XMT_RT:                                      ; Xmit routing message
                                289D  7193          $DISPATCH LPD$B_ETY(R6),TYPE=B,<- ; If we are an endnode,
                                289D  7194                    <ADJ$C_PTY_PH4N,110$>,- ; never send rtg messages
                                289D  7195                    <ADJ$C_PTY_PH3N,110$>>
                          0A    E0  28AC  7196          BBS     #LPD$V_BC,-             ; If broadcast circuit, always
                      70 22 A6        28AE  7197                  LPD$W_STS(R6),XMT_RT4   ; send Phase IV routing messages
                    50    1D A6    9A 28B1  7198          MOVZBL  LPD$B_ETY(R6),R0        ; Get our (adapted) "node type"
                04  0000014A'EF40    91 28B5  7199          CMPB    PTY_TO_PHASE[R0],#4     ; Are we supposed to be Phase IV?
                          62    13  28BD  7200          BEQL    XMT_RT4                 ; If so, go to Phase IV routine
                                28BF  7201          ;
                                28BF  7202          ;    Allocate and setup the buffer
                                28BF  7203          ;
                      51   5A A4    3C 28BF  7204          MOVZWL  RCB$W_MAX_ADDR(R4),R1   ; Get number of nodes
                         51   51    C0 28C3  7205          ADDL    R1,R1                   ; Need 1 word per entry
                         51   05    C0 28C6  7206          ADDL    #3+2,R1                 ; Add in header and trailer
                            0205    30 28C9  7207          BSBW    NET$DLL_QIO_CO          ; Call co-routine to allocate buffer
                   003C'C2   53    D0 28CC  7208          MOVL    R3,WQE$C_LENGTH+P1(R2)  ; Point to I/O buffer
                   0038'C2   53    CE 28D1  7209          MNEGL   R3,WQE$C_LENGTH+P2(R2)  ; Bias the I/O buffer length
                                28D6  7210          ;
                                28D6  7211          ;   Build the message
                                28D6  7212          ;
                             57   DD 28D6  7213          PUSHL   R7                      ; Save registers
                         83      07 90  28D8  7214          MOVB    #TR3C_MSG_RT,(R3)+      ; Enter type code
                             00   EF 28DB  7215          EXTZV   #TR4$V_ADDR_DEST,-      ; Get our address (without area)
                50   0E A4       0A 28DD  7216                  #TR4$S_ADDR_DEST,RCB$W_ADDR(R4),R0
                         83      50 B0  28E1  7217          MOVW    R0,(R3)+                ; Enter source node address
                             57   D4 28E4  7218          CLRL    R7                      ; Init check sum
                      50   5A A4    3C 28E6  7219          MOVZWL  RCB$W_MAX_ADDR(R4),R0   ; Get number of nodes
                         51   01    D0 28EA  7220          MOVL    #1,R1                   ; Init the node index
            58  00000100'EF41    B0 28ED  7221 50$:       MOVW    NET$AW_MIN_C_H[R1],R8   ; Get cost-hops to the node
                         83   58    B0 28F5  7222          MOVW    R8,(R3)+                ; Enter cost-hops to the node
                         57   58    A0 28F8  7223          ADDW    R8,R7                   ; Include in checksum
                         57   00    D8 28FB  7224          ADWC    #0,R7                   ; 1's complement add - needs
                                      28FE  7225                                          ; "end around carry"
                             51    D6 28FE  7226          INCL    R1                      ; Advance the node index
                            EA 50    F5 2900  7227          SOBGTR  R0,50$                  ; Loop for each node
                         83   57    B0 2903  7228          MOVW    R7,(R3)+                ; Enter the check sum
                              57 8ED0  2906  7229          POPL    R7                      ; Restore registers
                   0038'C2   53    C0 2909  7230          ADDL    R3,WQE$C_LENGTH+P2(R2)  ; Setup I/O buffer size
                         50   00'   D0 290E  7231          MOVL    S^#IO$_WRITELBLK,R0     ; Setup I/O fct code
```

```
                     2911  7232          CLRBIT  LPD$V_XMT_RT,-                ; No further need to send message
                     2911  7233                   LPD$B_XMTFLG(R6)            ;
              05     2915  7234          RSB                                   ; Return to co-routine to xmit
                     2916  7235
                     2916  7236 ;
                     2916  7237 ; We are an endnode.  Don't send routing messages.
                     2916  7238 ;
                     2916  7239
                     2916  7240 110$:   CLRBIT  LPD$V_XMT_RT,-                ; No further need to send message
                     2916  7241                   LPD$B_XMTFLG(R6)            ;
51    00    D0     291A  7242          MOVL    #LEV$C_NO_EVT,R1              ; No more events
50    01    D0     291D  7243          MOVL    #1,R0                         ; Allow state change
              05     2920  7244          RSB
```

```
                                2921  7246          .SBTTL  XMT_RT4 - Transmit a Phase IV routing message
                                2921  7247 ;+
                                2921  7248 ; XMT_RT4 - Transmit a Phase IV segmented routing message
                                2921  7249 ;
                                2921  7250 ; Inputs:
                                2921  7251 ;
                                2921  7252 ;       R11 = CRI CNR address
                                2921  7253 ;       R10 = CRI CNF address
                                2921  7254 ;       R7 = ADJ address
                                2921  7255 ;       R6 = LPD address
                                2921  7256 ;       R4 = RCB address
                                2921  7257 ;
                                2921  7258 ; Outputs:
                                2921  7259 ;
                                2921  7260 ;       R1 = Next event to be processed
                                2921  7261 ;       R0 = True if state change allowed, else false.
                                2921  7262 ;-
                                2921  7263 XMT_RT4:
                     53 A6  95  2921  7264          TSTB    LPD$B_SRM_LEFT(R6)      ; Any bits left to check?
                        0B  12  2924  7265          BNEQ    5$                      ; Branch if so
                                2926  7266 3$:      CLRBIT  LPD$V_XMT_RT,LPD$B_XMTFLG(R6) ; Indicate "transmission" done
                  51    00  D0  292A  7267          MOVL    #LEV$C_NO_EVT,R1        ; No more events
                  50    01  D0  292D  7268          MOVL    #1,R0                   ; Allow state change
                            05  2930  7269          RSB
                                2931  7270 5$:      ;
                                2931  7271          ;   Allocate and setup the buffer
                                2931  7272          ;
                59    06 A7  3C  2931  7273          MOVZWL  ADJ$W_BUFSIZ(R7),R9     ; Get adjacent node's buffer size
                           04  12  2935  7274          BNEQ    8$                      ; Branch if "known"
                59    50 A6  3C  2937  7275          MOVZWL  LPD$W_BUFSIZ(R6),R9     ; (& should never get here)
                                2938  7276                                          ; Else, use our own buffer size
                   51    59  D0  2938  7277 8$:      MOVL    R9,R1                   ; Indicate size of extra buffer
                      0190  30  293B  7278          BSBW    NET$DLL_QIO_CO          ; Call co-routine to allocate buffer
             003C'C2    53  D0  293E  7279          MOVL    R3,WQE$C_LENGTH+P1(R2)  ; Point to I/O buffer
             0038'C2    53  CE  2943  7280          MNEGL   R3,WQE$C_LENGTH+P2(R2)  ; Bias the I/O buffer size
             58    59   06  C3  2948  7281          SUBL3   #6,R9,R8                ; Subtract out required overhead
          58   00000044 8F  C6  294C  7282          DIVL    #4+<2*LPD$C_SRM_NODES>,R8 ; Compute number of segments
                                2953  7283                                          ; which neighbor can handle in 1 packet
                   83    07  90  2953  7284          MOVB    #TR4C_MSG_RT,(R3)+      ; Enter type code
                83    0E A4  B0  2956  7285          MOVW    RCB$W_ADDR(R4),(R3)+    ; Enter source node address
                         83  94  295A  7286          CLRB    (R3)+                   ; Skip reserved byte
                         53  DD  295C  7287          PUSHL   R3                      ; Save address of first segment
                                2958  7288          ;
                                2958  7289          ;   For each segment with it's bit set in the SRM bitmask,
                                2958  7290          ;   copy the associated cost/hops entries from the cost/hops matrix
                                2958  7291          ;   into the message.  Make special provisions so that node numbers
                                2958  7292          ;   less than 1, and greater than MAX ADDRESS are not sent.
                                2958  7293          ;
                                2958  7294          ASSUME  LPD$C_SRM_SIZE EQ 32
                   E0 8F  8A  2958  7295 10$:      BICB    #^C<LPD$C_SRM_SIZE-1>,- ; Make sure index is always a modulo
                      52 A6       295B  7296                  LPD$B_SRM_POS(R6)       ; of the bitmask size (wrap around)
                50    52 A6  9A  295D  7297          MOVZBL  LPD$B_SRM_POS(R6),R0    ; Get current position in SRM bitmask
                      52 A6  96  2961  7298          INCB    LPD$B_SRM_POS(R6)       ; Update current segment number
             43 5A A6    50  E5  2964  7299          BBCC    R0,LPD$G_XMT_SRM(R6),30$ ; Skip if segment not to be sent
                                2969  7300          CLRBIT  R0,LPD$G_SRM(R6)        ; Optimize next pass; already done
                50    50   05  78  296C  7301          ASHL    #LPD$C_SRM_SHFT,R0,R0   ; Compute starting node address
                   5A A4   50  B1  2970  7302          CMPW    R0,RCB$W_MAX_ADDR(R4)   ; Higher than MAX ADDRESS?
```

```
                  34   1A  297F  7303          BGTRU   30$                      ; If so, do not send
        51   50   20   C1  2981  7304          ADDL3   #LPD$C_SRM_NODES,R0,R1   ; Compute ending+1 node address
                  51   D7  2985  7305          DECL    R1                       ; Compute ending node address
          5A A4   51   B1  2987  7306          CMPW    R1,RCB$W_MAX_ADDR(R4)    ; Ending address < MAX ADDRESS ?
                  04   1B  298B  7307          BLEQU   20$                      ; Branch if ok
        51   5A A4   3C  298D  7308          MOVZWL  RCB$W_MAX_ADDR(R4),R1    ; Never send > MAX ADDRESS
             51   50   C2  2991  7309 20$:     SUBL    R0,R1                    ; Compute number of nodes-1 in segment
                  51   D6  2994  7310          INCL    R1                       ; Compute number of nodes in segment
                  1D   13  2996  7311          BEQL    30$                      ; Branch if nothing to send
             83   51   B0  2998  7312          MOVW    R1,(R3)+                 ; Set number of nodes in segment
             83   50   B0  299B  7313          MOVW    R0,(R3)+                 ; Set starting node address
                  34   BB  299E  7314          PUSHR   #^M<R2,R4,R5>            ; Save registers
   50   00000100'EF40   3E  29A0  7315          MOVAW   NET$AW_MIN_C_H[R0],R0    ; Get address of cost/hops entries
             51   02   C4  29A8  7316          MULL    #2,R1                    ; Compute number of bytes in segment
        63   60   51   28  29AB  7317          MOVC    R1,(R0),(R3)             ; Store cost/hops entries in msg
                  34   BA  29AF  7318          POPR    #^M<R2,R4,R5>            ; Restore registers
                  58   D7  29B1  7319          DECL    R8                       ; Indicate segment filled
                  05   13  29B3  7320          BEQL    35$                      ; Branch if cannot fit any more segments
             53 A6   97  29B5  7321 30$:     DECB    LPD$B_SRM_LEFT(R6)       ; Decrement number of bits left to check
                  A7   14  29B8  7322          BGTR    10$                      ; Loop through all segments
                           29BA  7323          ;
                           29BA  7324          ;   Compute checksum on all segments in message, and store it
                           29BA  7325          ;
                  50 8ED0  29BA  7326 35$:     POPL    R0                       ; Get address of first segment
             53   50   D1  29BD  7327          CMPL    R0,R3                    ; Any segments at all?
                  1A   13  29C0  7328          BEQL    70$                      ; Branch if not
             51   01   D0  29C2  7329          MOVL    #1,R1                    ; Init checksum
             51   80   A0  29C5  7330 40$:     ADDW    (R0)+,R1                 ; Add to 1's complement checksum
             51   00   D8  29C8  7331          ADWC    #0,R1                    ; add "end around carry"
             53   50   D1  29CB  7332          CMPL    R0,R3                    ; At end of message?
                  F5   1F  29CE  7333          BLSSU   40$                      ; Continue if not
             83   51   B0  29D0  7334          MOVW    R1,(R3)+                 ; Store checksum
                           29D3  7335          ;
                           29D3  7336          ;   Send the message
                           29D3  7337          ;
        0038'C2   53   C0  29D3  7338          ADDL    R3,WQE$C_LENGTH+P2(R2)   ; Set I/O buffer size
             50   00'  D0  29D8  7339          MOVL    S^#IO$_WRITELBLK,R0      ; Set I/O function code
                  05   29DB  7340          RSB                              ; Return to issue I/O
                           29DC  7341
                           29DC  7342          ;
                           29DC  7343          ;   No segments in message.  Do not send anything.
                           29DC  7344          ;
                  50   D4  29DC  7345 70$:     CLRL    R0                       ; Do not issue I/O
                  9E   16  29DE  7346          JSB     @(SP)+                   ; Return to abort co-routine
                FF43   31  29E0  7347          BRW     3$                       ; Indicate we are done
```

```
                                29E3  7349          .SBTTL  XMT_ART - Transmit a Phase IV area routing message
                                29E3  7350 ;+
                                29E3  7351 ; XMT_ART - Transmit a Phase IV segmented area routing message
                                29E3  7352 ;
                                29E3  7353 ; Inputs:
                                29E3  7354 ;
                                29E3  7355 ;       R11 = CRI CNR address
                                29E3  7356 ;       R10 = CRI CNF address
                                29E3  7357 ;       R7 = ADJ address
                                29E3  7358 ;       R6 = LPD address
                                29E3  7359 ;       R4 = RCB address
                                29E3  7360 ;
                                29E3  7361 ; Outputs:
                                29E3  7362 ;
                                29E3  7363 ;       R1 = Next event to be processed
                                29E3  7364 ;       R0 = True if state change allowed, else false.
                                29E3  7365 ;-
                                29E3  7366 XMT_ART:
          55 A6    95           29E3  7367          TSTB    LPD$B_ASRM_LEFT(R6)     ; Any bits left to check?
             0C    12           29E6  7368          BNEQ    5$                      ; Branch if so
                                29E8  7369 3$:      CLRBIT  LPD$V_XMT_ART,LPD$B_XMTFLG(R6) ; Indicate "transmission" done
       51    00    D0           29ED  7370          MOVL    #LEV$C_NO_EVT,R1        ; No more events
       50    01    D0           29F0  7371          MOVL    #1,R0                   ; Allow state change
                   05           29F3  7372          RSB
                                29F4  7373 5$:      ;
                                29F4  7374          ;   Allocate and setup the buffer
                                29F4  7375          ;
    59    06 A7    3C           29F4  7376          MOVZWL  ADJ$W_BUFSIZ(R7),R9     ; Get adjacent node's buffer size
             04    12           29F8  7377          BNEQ    8$                      ; Branch if "known"
    59    50 A6    3C           29FA  7378          MOVZWL  LPD$W_BUFSIZ(R6),R9     ; (& should never get here)
                                29FE  7379                                          ; Else, use our own buffer size
          51    59    D0       29FE  7380 8$:      MOVL    R9,R1                   ; Indicate size of extra buffer
                00CD    30     2A01  7381          BSBW    NET$DLL_QIO_CO          ; Call co-routine to allocate buffer
       003C'C2    53    D0     2A04  7382          MOVL    R3,WQE$C_LENGTH+P1(R2)  ; Point to I/O buffer
       0038'C2    53    CE     2A09  7383          MNEGL   R3,WQE$C_LENGTH+P2(R2)  ; Bias the I/O buffer size
       58    59    06    C3     2A0E  7384          SUBL3   #6,R9,R8                ; Subtract out required overhead
    58   00000084 8F    C6     2A12  7385          DIVL    #4+<2*LPD$C_ASRM_AREAS>,R8 ; Compute number of segments
                                2A19  7386                                          ; which neighbor can handle in 1 packet
             83    09    90     2A19  7387          MOVB    #TR4C_MSG_ART,(R3)+     ; Enter type code
          83    0E A4    B0     2A1C  7388          MOVW    RCB$W_ADDR(R4),(R3)+    ; Enter source area address
                83    94        2A20  7389          CLRB    (R3)+                   ; Skip reserved byte
                53    DD        2A22  7390          PUSHL   R3                      ; Save address of first segment
                                2A24  7391          ;
                                2A24  7392          ;   For each segment with it's bit set in the SRM bitmask,
                                2A24  7393          ;   copy the associated cost/hops entries from the cost/hops matrix
                                2A24  7394          ;   into the message.  Make special provisions so that node numbers
                                2A24  7395          ;   less than 1, and greater than MAX ADDRESS are not sent.
                                2A24  7396          ;
                                2A24  7397          ASSUME  LPD$C_ASRM_SIZE EQ 1    ; && fix this
             FF 8F    8A        2A24  7398 10$:     BICB    #^C<LPD$C_ASRM_SIZE-1>,- ; Make sure index is always a modulo
             54 A6              2A27  7399                  LPD$B_ASRM_POS(R6)      ; of the bitmask size (wrap around)
       50    54 A6    9A        2A29  7400          MOVZBL  LPD$B_ASRM_POS(R6),R0   ; Get current position in SRM bitmask
             54 A6    96        2A2D  7401          INCB    LPD$B_ASRM_POS(R6)      ; Update current segment number
    50 62 A6    50    E5        2A30  7402          BBCC    R0,LPD$G_XMT_ASRM(R6),30$ ; Skip if segment not to be sent
                               2A35  7403          CLRBIT  R0,LPD$G_ASRM(R6)       ; Optimize next pass; already done
       50    50    06    78     2A3A  7404          ASHL    #LPD$C_ASRM_SHFT,R0,R0  ; Compute starting area address
       008C C4    50    91     2A3E  7405          CMPB    R0,RCB$B_MAX_AREA(R4)   ; Higher than MAX AREA?
```

```
                    40   1A  2A43  7406          BGTRU   30$                          ; If so, do not send
 51   50   00000040 8F   C1  2A45  7407          ADDL3   #LPD$C_ASRM_AREAS,R0,R1      ; Compute ending+1 area address
                    51   D7  2A4D  7408          DECL    R1                           ; Compute ending area address
                    50   D5  2A4F  7409          TSTL    R0                           ; Starting address > 0 ?
                    02   1A  2A51  7410          BGTRU   15$                          ; Branch if ok
                    50   D6  2A53  7411          INCL    R0                           ; Never send area #0
          008C C4   51   91  2A55  7412 15$:     CMPB    R1,RCB$B_MAX_AREA(R4)        ; Ending address < MAX AREA ?
                    05   1B  2A5A  7413          BLEQU   20$                          ; Branch if ok
          51   008C C4   9A  2A5C  7414          MOVZBL  RCB$B_MAX_AREA(R4),R1        ; Never send > MAX AREA
               51   50   C2  2A61  7415 20$:     SUBL    R0,R1                        ; Compute number of nodes-1 in segment
                    51   D6  2A64  7416          INCL    R1                           ; Compute number of nodes in segment
                    1D   13  2A66  7417          BEQL    30$                          ; Branch if nothing to send
               83   51   B0  2A68  7418          MOVW    R1,(R3)+                     ; Set number of nodes in segment
               83   50   B0  2A6B  7419          MOVW    R0,(R3)+                     ; Set starting node address
                    34   BB  2A6E  7420          PUSHR   #^M<R2,R4,R5>                ; Save registers
     50   00000900'EF40   3E  2A70  7421          MOVAW   NET$AW_AREA_C_H[R0],R0       ; Get address of cost/hops entries
               51   02   C4  2A78  7422          MULL    #2,R1                        ; Compute number of bytes in segment
          63   60   51   28  2A7B  7423          MOVC    R1,(R0),(R3)                 ; Store cost/hops entries in msg
                    34   BA  2A7F  7424          POPR    #^M<R2,R4,R5>                ; Restore registers
                    58   D7  2A81  7425          DECL    R8                           ; Indicate segment filled
                    05   13  2A83  7426          BEQL    35$                          ; Branch if cannot fit any more segments
                 55 A6   97  2A85  7427 30$:     DECB    LPD$B_ASRM_LEFT(R6)          ; Decrement number of bits left to check
                    9A   14  2A88  7428          BGTR    10$                          ; Loop through all segments
                             2A8A  7429          ;
                             2A8A  7430          ;   Compute checksum on all segments in message, and store it
                             2A8A  7431          ;
                    50 8ED0  2A8A  7432 35$:     POPL    R0                           ; Get address of first segment
               53   50   D1  2A8D  7433          CMPL    R0,R3                        ; Any segments at all?
                    1A   13  2A90  7434          BEQL    70$                          ; Branch if not
               51   01   D0  2A92  7435          MOVL    #1,R1                        ; Init checksum
               51   80   A0  2A95  7436 40$:     ADDW    (R0)+,R1                     ; Add to 1's complement checksum
               51   00   D8  2A98  7437          ADWC    #0,R1                        ; add "end around carry"
               53   50   D1  2A9B  7438          CMPL    R0,R3                        ; At end of message?
                    F5   1F  2A9E  7439          BLSSU   40$                          ; Continue if not
               83   51   B0  2AA0  7440          MOVW    R1,(R3)+                     ; Store checksum
                             2AA3  7441          ;
                             2AA3  7442          ;   Send the message
                             2AA3  7443          ;
          0038'C2   53   C0  2AA3  7444          ADDL    R3,WQE$C_LENGTH+P2(R2)       ; Set I/O buffer size
               50   00'  D0  2AA8  7445          MOVL    S^#IO$_WRITELBLK,R0          ; Set I/O function code
                         05  2AAB  7446          RSB                                  ; Return to issue I/O
                             2AAC  7447
                             2AAC  7448          ;
                             2AAC  7449          ;   No segments in message.  Do not send anything.
                             2AAC  7450          ;
                    50   D4  2AAC  7451 70$:     CLRL    R0                           ; Do not issue I/O
                    9E   16  2AAE  7452          JSB     @(SP)+                       ; Return to abort co-routine
                  FF35   31  2AB0  7453          BRW     3$                           ; Indicate we are done
```

```
                        2AB3          .SBTTL  CHK_IO - Check for multiple transmits
                        2AB3  ;+
                        2AB3  ; CHK_IO - See if its okay to transmit a message
                        2AB3  ;
                        2AB3  ; This routine ensures that only 1 transmit is outstanding at a time.
                        2AB3  ; This restriction applies only to non-broadcast circuits.
                        2AB3  ;
                        2AB3  ; Inputs:       R6 = LPD address
                        2AB3  ;
                        2AB3  ; Outputs:      R0 = True if ok to send
                        2AB3  ;                    False if transmit is outstanding - cannot transmit
                        2AB3  ;-
                        2AB3  CHK_IO:                                   ; See if its okay to xmit
            00    E1    2AB3          BBC     #LPD$V_ACTIVE,-           ; If circuit is no longer active,
      16 22 A6          2AB5                      LPD$W_STS(R6),70$     ; do not allow the I/O
         50 01    D0    2AB8          MOVL    #1,R0                     ; Assume its okay to xmit
  08 22 A6   0A    E1    2ABB          BBC     #LPD$V_BC,LPD$W_STS(R6),20$ ; If broadcast circuit, its ok
                        2AC0          ;
                        2AC0          ;   Allow up to 20 transmits at a time for NI circuits
                        2AC0          ;
   15    1B A6    91    2AC0          CMPB    LPD$B_ASTCNT(R6),#21      ; Is AST queue getting too big?
            0A    1B    2AC4          BLEQU   90$                       ; If not, allow it
            06    11    2AC6          BRB     70$                       ; Else, disallow it
                        2AC8          ;
                        2AC8          ;   Allow only 1 transmit at a time for point-to-point circuits
                        2AC8          ;
   01    1B A6    91    2AC8  20$:    CMPB    LPD$B_ASTCNT(R6),#1       ; Is AST queue getting too big?
                        2ACC                                            ; (ASTCNT=1 if no transmits are active,
                        2ACC                                            ;  the 1 is for the active receiver)
            02    1B    2ACC          BLEQU   90$                       ; If not, allow it
            50    D4    2ACE  70$:    CLRL    R0                        ; Message cannot be sent
                  05    2AD0  90$:    RSB                               ; Done
```

```
                                        2AD1  7488          .SBTTL  NET$DLL_QIO_CO - Common QIO routine
                                        2AD1  7489 ;+
                                        2AD1  7490 ; NET$DLL_QIO_CO        - Common co-routine to issue a DLL QIO
                                        2AD1  7491 ;
                                        2AD1  7492 ; Inputs:       R6 = LPD address
                                        2AD1  7493 ;               R1 = Maximum size of optional buffer needed for QIO
                                        2AD1  7494 ;
                                        2AD1  7495 ; Outputs:      R1      Next "event longword" to be processed
                                        2AD1  7496 ;               R0      Low bit set if state change is permitted,
                                        2AD1  7497 ;                       Low bit clear to avoid state change
                                        2AD1  7498 ;
                                        2AD1  7499 ;               R2-R4 are destroyed.
                                        2AD1  7500 ;
                                        2AD1  7501 ; This routine makes a co-routine call back to the caller after it sets
                                        2AD1  7502 ; up the WQE for the QIO.  On return from the co-routine call, this routine
                                        2AD1  7503 ; will issue the QIO and cause the appropriate event transition to be taken.
                                        2AD1  7504 ;
                                        2AD1  7505 ; Input to co-routine:
                                        2AD1  7506 ;
                                        2AD1  7507 ;               R2 = WQE address
                                        2AD1  7508 ;               R3 = Pointer to optional QIO buffer (if any)
                                        2AD1  7509 ;               R6 = LPD address
                                        2AD1  7510 ;
                                        2AD1  7511 ;               R4-R5,R7-R11 contain original values.
                                        2AD1  7512 ;
                                        2AD1  7513 ; Output from co-routine:
                                        2AD1  7514 ;
                                        2AD1  7515 ;               R0 = Function code for QIO
                                        2AD1  7516 ;-
                        00000000        2AD1  7517          IOSB =   0                          ; Define WQE extensions to hold the I/O
                        00000008        2AD1  7518          P5   =   8                          ; status block and the QIO parameters
                        0000000C        2AD1  7519          P4   =  12                          ;
                        00000010        2AD1  7520          P3   =  16                          ;
                        00000014        2AD1  7521          P2   =  20                          ;
                        00000018        2AD1  7522          P1   =  24                          ;
                        0000001C        2AD1  7523          FUNC =  28                          ; I/O function (word)
                        00000020        2AD1  7524          IOWQE_LENGTH = 32                   ; Size of extension (longword aligned)
                                        2AD1  7525                                              ;
                                        2AD1  7526 NET$DLL_QIO_CO:                              ; Common DLL Qio co-routine
                  50    03    D0        2AD1  7527          MOVL    #WQE$C_SUB_AST,R0           ; Indicate WQE subtype
                  51    22    C0        2AD4  7528          ADDL    #IOWQE_LENGTH+2,R1          ; Add in WQE I/O extension
                                        2AD7  7529                                              ; Add 2 bytes in case CRC16 needed (X25)
                     D526'    30        2AD7  7530          BSBW    WQE$ALLOCATE                ; Allocate the element - always succeeds
                  20 A6       B0        2ADA  7531          MOVW    LPD$W_PTH(R6),-             ;
                  12 A2                 2ADD  7532              WQE$W_REQIDT(R2)                ; Setup path i.d.
                     16       9B        2ADF  7533          MOVZBW  S^#LEV$C_IO_SUCC,-          ; Setup default QIO success event
                  10 A2                 2AE1  7534              WQE$B_EVT(R2)                   ;
                     15       D0        2AE3  7535          MOVL    #LEV$C_IO_FAIL,-            ; Setup default QIO failure event
                  14 A2                 2AE5  7536              WQE$L_PM2(R2)                   ;
0C A2       00002BE0'EF       9E        2AE7  7537          MOVAB   QIOAST,WQE$L_ACTION(R2)     ; Setup post processing routine
                  53    24 A2       9E  2AEF  7538          MOVAB   WQE$C_LENGTH+IOSB(R2),R3    ; Get start WQE extension
                        83       7C   2AF3  7539          CLRQ    (R3)+                       ; Zero the IOSB image
                        83       7C   2AF5  7540          CLRQ    (R3)+                       ; Zero P5 and P4
                        83       7C   2AF7  7541          CLRQ    (R3)+                       ; Zero P3 and P2
                        83       D4   2AF9  7542          CLRL    (R3)+                       ; Zero P1
                08 22 A6        0A   E1  2AFB  7543          BBC     #LPD$V_BC,LPD$W_STS(R6),10$ ; Skip if non-broadcast driver
2C A2       00000100'EF       9E        2B00  7544          MOVAB   NET$G_ALL_ROU,WQE$C_LENGTH+P5(R2) ; Set P5 = "all routers"
```

```
                53   46 A2    9E   2B08  7545 10$:    MOVAB   WQE$C_LENGTH+IOWQE_LENGTH+2(R2),R3 ; Point to optional buffer
                           9E   16   2B0C  7546         JSB     @(SP)+                          ; Get QIO data
                40 A2      50   B0   2B0E  7547         MOVW    R0,WQE$C_LENGTH+FUNC(R2); Store I/O function code
                           03   12   2B12  7548         BNEQ    15$                             ; Br if function supplied
                      00B7      31   2B14  7549         BRW     200$                            ; Caller bailed out of I/O
                                     2B17  7550 15$:     ;
                                     2B17  7551         ;   If we are transmitting an X.25 datagram, then calculate the CRC-16
                                     2B17  7552         ;   and append it to the front of the buffer.
                                     2B17  7553         ;
             23 22 A6     07   E1   2B17  7554         BBC     #LPD$V_X25,LPD$W_STS(R6),20$ ; Skip if not X.25 datalink
              0000'8F     50   B1   2B1C  7555         CMPW    R0,#IO$_WRITELBLK               ; Writing a datagram?
                          1C   12   2B21  7556         BNEQ    20$                             ; Branch if not
                          52   DD   2B23  7557         PUSHL   R2                              ; Save WQE address
          00   00000106'EF    0B   2B25  7558         CRC     CRC16,#0,-                      ; Calculate CRC16 checksum
                       38 A2       2B2C  7559                 WQE$C_LENGTH+P2(R2),-
                       3C B2       2B2E  7560                 @WQE$C_LENGTH+P1(R2)
                          52 8ED0  2B30  7561         POPL    R2                              ; Restore WQE address
                    38 A2     02   C0   2B33  7562         ADDL    #2,WQE$C_LENGTH+P2(R2)  ; Account for CRC16 in length
                    3C A2     02   C2   2B37  7563         SUBL    #2,WQE$C_LENGTH+P1(R2)  ; Move back message pointer
                    3C B2     50   B0   2B3B  7564         MOVW    R0,@WQE$C_LENGTH+P1(R2) ; Append to front of msg
                                     2B3F  7565 20$:     ;
                                     2B3F  7566         ;   If this is a write request, then journal the data
                                     2B3F  7567         ;
                       40 A2     B1   2B3F  7568         CMPW    WQE$C_LENGTH+FUNC(R2),- ; Write request?
                    0000'8F          2B42  7569                 #IO$_WRITELBLK
                          1D   12   2B45  7570         BNEQ    25$                             ; If so,
                       D4B6'   30   2B47  7571         BSBW    NET$JNX_CO                      ; Initialize journalling co-routine
                       17 50   E9   2B4A  7572         BLBC    R0,25$                          ; Skip if journalling not enabled
                          81   94   2B4D  7573         CLRB    (R1)+                           ; Record type = start of transmit
                    81    20 A6   90   2B4F  7574         MOVB    LPD$B_PTH_INX(R6),(R1)+ ; LPD index
                    81    38 A2   B0   2B53  7575         MOVW    WQE$C_LENGTH+P2(R2),(R1)+ ; Length of message
                          38 A2   2C   2B57  7576         MOVC5   WQE$C_LENGTH+P2(R2),-   ; Store data into journal record
                          3C B2       2B5A  7577                 @WQE$C_LENGTH+P1(R2),-
                61    34    00        2B5C  7578                 #0,#64-12,(R1)
                      51    53   D0   2B5F  7579         MOVL    R3,R1                           ; Set ending address of record
                            9E   16   2B62  7580         JSB     @(SP)+                          ; Log the journal record
                                     2B64  7581 25$:     ;
                                     2B64  7582         ;   Queue the I/O
                                     2B64  7583         ;
                                     2B64  7584         $QIO_S  -                               ; Issue QIO
                                     2B64  7585             FUNC = WQE$C_LENGTH+FUNC(R2),-
                                     2B64  7586             EFN  = #NET$C_EFN_ASYN,-
                                     2B64  7587             CHAN = LPD$W_CHAN(R6),-
                                     2B64  7588             IOSB = WQE$C_LENGTH+IOSB(R2),-
                                     2B64  7589             P5   = WQE$C_LENGTH+P5(R2),-
                                     2B64  7590             P4   = WQE$C_LENGTH+P4(R2),-
                                     2B64  7591             P3   = WQE$C_LENGTH+P3(R2),-
                                     2B64  7592             P2   = WQE$C_LENGTH+P2(R2),-
                                     2B64  7593             P1   = @WQE$C_LENGTH+P1(R2),-; $QIO_S macro does a PUSHAB for P1
                                     2B64  7594             ASTADR = B^NET$DLLQIOAST,-      ;
                                     2B64  7595             ASTPRM = R2                     ; Use the WQE ptr as parameter
                      1B A6   96     2B8E  7596         INCB    LPD$B_ASTCNT(R6)                ; Account for $QIO
                53    50      D0     2B91  7597         MOVL    R0,R3                           ; Save I/O status
                    D469'     30     2B94  7598         BSBW    NET$JNX_CO                      ; Initialize journalling co-routine
                      10 50   E9     2B97  7599         BLBC    R0,30$                          ; Branch if journalling not enabled
                81    11      90     2B9A  7600         MOVB    #^X11,(R1)+                     ; Journal record type = QIO
              81    20 A6     90     2B9D  7601         MOVB    LPD$B_PTH_INX(R6),(R1)+ ; LPD index
```

```
                81   40 A2    B0  2BA1  7602          MOVW    WQE$C_LENGTH+FUNC(R2),(R1)+ ; I/O function code
                   81   53    D0  2BA5  7603          MOVL    R3,(R1)+                    ; Status from QIO request
                        9E    16  2BA8  7604          JSB     @(SP)+                      ; Log the journalling record
                     10 53    E9  2BAA  7605 30$:     BLBC    R3,50$                      ; Br if QIO request failed
                                  2BAD  7606          ;
                                  2BAD  7607          ;   I/O queued.  Set timer and wait for AST
                                  2BAD  7608          ;
    53  00000000 6B49D200 8F    7D  2BAD  7609          MOVQ    #TR$C_TIM_DLLIO*-           ; Setup I/O timer interval
                                  2BB8  7610                  10*1000*1000,R3             ; in quadword VMS clock ticks
                      0065    30  2BB8  7611          BSBW    SET_IOTIM                   ; Cancel old timer, set new one
                        0A    11  2BBB  7612          BRB     100$                        ; Continue
                                  2BBD  7613          ;
                                  2BBD  7614          ;   I/O failure.  Setup status and queue WQE
                                  2BBD  7615          ;
                24 A2   53    B0  2BBD  7616 50$:     MOVW    R3,WQE$C_LENGTH+IOSB(R2); Store status in IOSB field
                   50   52    D0  2BC1  7617          MOVL    R2,R0                       ; Get the WQE address
                      D439'   30  2BC4  7618          BSBW    WQE$INSQUE                  ; Queue it
                   51   00    D0  2BC7  7619 100$:    MOVL    S^#LEV$C_NO_EVT,R1          ; Setup next event longword
                   50   01    90  2BCA  7620          MOVB    #1,R0                       ; Allow state change
                             05  2BCD  7621          RSB
                                  2BCE  7622
                                  2BCE  7623 200$:                                      ; Caller doesn't want to issue I/O
                   50   52    D0  2BCE  7624          MOVL    R2,R0                       ; Set the WQE address
                      D42C'   30  2BD1  7625          BSBW    NET$DEALLOCATE              ; Deallocate it
                        F1    11  2BD4  7626          BRB     100$                        ; and return success
                                  2BD6  7627
                                  2BD6  7628 NET$DLLQIOAST:
                         0000     2BD6  7629          .WORD   0                           ; No need to save regs
                50   04 AC    D0  2BD8  7630          MOVL    4(AP),R0                    ; Get the WQE address
                      D421'   30  2BDC  7631          BSBW    WQE$INSQUE                  ; Queue it
                             04  2BDF  7632          RET
                                  2BE0  7633
              51   10 A5    D0  2BE0  7634 QIOAST:  MOVL    WQE$W_REQIDT-2(R5),R1       ; Put Path i.d. into high order word
            51   0114 8F    B0  2BE4  7635          MOVW    #<<WQE$C_QUAL_DLL>@8>!-     ; Setup timer qualifier
                                  2BE9  7636                      LEV$C_IO_TIMOUT,R1      ; and timer event
                      D414'   30  2BE9  7637          BSBW    WQE$CANCEL_TIM              ; Cancel the timer
                      E25B    30  2BEC  7638          BSBW    FIND_WQE_CTX                ; Locate CNF, LPD, ADJ blocks
                   26 50      E9  2BEF  7639          BLBC    R0,230$                     ; If LPD no longer exists, skip event
                   1B A6      97  2BF2  7640          DECB    LPD$B_ASTCNT(R6)            ; Account for AST
                      D408'   30  2BF5  7641          BSBW    NET$JNX_CO                  ; Initialize journalling co-routine
                   11 50      E9  2BF8  7642          BLBC    R0,30$                      ; Branch if journalling not enabled
                81    22      90  2BFB  7643          MOVB    #^X22,(R1)+                 ; Journal record type = QIO AST
             81    20 A6      90  2BFE  7644          MOVB    LPD$B_PTH_INX(R6),(R1)+     ; LPD index
             81    40 A5      B0  2C02  7645          MOVW    WQE$C_LENGTH+FUNC(R5),(R1)+ ; I/O function code
             81    24 A5      7D  2C06  7646          MOVQ    WQE$C_LENGTH+IOSB(R5),(R1)+ ; I/O completion status
                      9E      16  2C0A  7647          JSB     @(SP)+                      ; Log the journalling record
          05 24 A5    E8  2C0C  7648 30$:     BLBS    WQE$C_LENGTH+IOSB(R5),220$  ; If LBC then I/O failed
10 A5     14 A5      90  2C10  7649          MOVB    WQE$L_PM2(R5),WQE$B_EVT(R5) ; Set failure event
                      E17B    30  2C15  7650 220$:    BSBW    PROC_EVT                    ; Process the event
                      E16D    30  2C18  7651 230$:    BSBW    KILL_WQE                    ; Deallocate the WQE
                             05  2C1B  7652          RSB
                                  2C1C  7653
                                  2C1C  7654 500$:    BUG_CHECK  NETNOSTATE,FATAL         ; Signal the bug
```

```
                                2C20  7656          .SBTTL  SET_IOTIM - Set I/O timer
                                2C20  7657 ;+
                                2C20  7658 ; SET_IOTIM - Set I/O timer
                                2C20  7659 ;
                                2C20  7660 ; INPUTS:        R6      LPD ptr
                                2C20  7661 ;                R3/R4   Quadword value of timer
                                2C20  7662 ;
                                2C20  7663 ; OUTPUTS:       R5-R11  Preserved
                                2C20  7664 ;-
                                2C20  7665 SET_IOTIM:                                   ; Start the I/O timer
50    0114 8F    B0             2C20  7666          MOVW    #<<WQE$C_QUAL_DLL>@8>!-     ; Setup timer qualifier
                                2C25  7667              LEV$C_IO_TIMOUT,R0              ; and timer event
   51   20 A6    B0             2C25  7668          MOVW    LPD$W_PTR(R6),R1            ; Get LPD index
51    51   10    78             2C29  7669          ASHL    #16,R1,R1                   ; Shift into upper word (REQIDT)
      51   50    B0             2C2D  7670          MOVW    R0,R1                       ;
52    E14C CF    9E             2C30  7671          MOVAB   NET$DLL_PRC_WQE,R2          ; Setup action routine address
         D3C8'   30             2C35  7672          BSBW    WQE$RESET_TIM               ; Reset the timer
                 05             2C38  7673          RSB
```

```
                    2C39          .SBTTL  RESET_CHAN - Cancel all device I/O
                    2C39  ;+
                    2C39  ; RESET_CHAN    - Cancel all the I/O queued to device.
                    2C39  ;
                    2C39  ; FUNCTIONAL DESCRIPTION:
                    2C39  ;
                    2C39  ; If a channel is active to the driver then call the driver to cancel ALL
                    2C39  ; the I/O on the device.  A $CANCEL is not sufficient since the PID field
                    2C39  ; of the internal IRPs queued to the data link driver by NETDRIVER would
                    2C39  ; not match hence not all of the packets would be cancelled.
                    2C39  ;
                    2C39  ; INPUTS:       R11     CRI CNR pointer
                    2C39  ;               R10     CRI CNF pointer
                    2C39  ;               R6      LPD pointer
                    2C39  ;
                    2C39  ; OUTPUTS:      R0      Status
                    2C39  ;
                    2C39  ;               All registers are preserved
                    2C39  ;-
                    2C39  RESET_CHAN:
                    2C39          $CANCEL_S CHAN = LPD$W_CHAN(R6) ; Cancel stuff on the queue
50    01    D0      2C44          MOVL    #1,R0                   ; Return success
            05      2C47          RSB
```

```
                                2C48          .SBTTL  NET$GET_LPD_CRI - Locate CNF given LPD index
                                2C48  ;+
                                2C48  ; NET$GET_LPD_CRI - Locate CNR and CNF given LPD index
                                2C48  ;
                                2C48  ; INPUTS        R11-R9  Scratch
                                2C48  ;               R8      Low byte contains LPD index
                                2C48  ;               R7,R6   Scratch
                                2C48  ;
                                2C48  ; OUTPUTS:      R11     CNR address
                                2C48  ;               R10     CNF address
                                2C48  ;               R9-R7   Garbage
                                2C48  ;               R6      LPD if low bit set in R0
                                2C48  ;               R0      Low bit set if successful
                                2C48  ;                       Low bit clear otherwise
                                2C48  ;-
                                2C48  NET$GET_LPD_CRI::
5B   00000000'EF    D0          2C48          MOVL    NET$GL_CNR_CRI,R11      ; Get data base root for CRI
                5A  D4          2C4F          CLRL    R10                     ; No CNF yet
                4D  10          2C51          BSBB    NET$FIND_LPD            ; Find the LPD via index in R8
             1B 50  E9          2C53          BLBC    R0,10$                  ; If LPD then none
          50    5B  D0          2C56          MOVL    R11,R0                  ; Make a copy
       58    20 A6  3C          2C59          MOVZWL  LPD$W_PTH(R6),R8        ; Get full LPD path i.d.
          50    60  D0          2C5D  5$:     MOVL    CNF$L_FLINK(R0),R0      ; Get next CNF
          5B    50  D1          2C60          CMPL    R0,R11                  ; At head of list?
                0C  13          2C63          BEQL    10$                     ; If EQL yes, return with LBC in R0
       12 A0    58  B1          2C65          CMPW    R8,CNF$W_ID(R0)         ; This it?
                F2  12          2C69          BNEQ    5$                      ; If NEQ keep trying
          5A    50  D0          2C6B          MOVL    R0,R10                  ; Copy CNF address
          50    00' D0          2C6E          MOVL    S^#SS$_NORMAL,R0        ; Set status
                    05          2C71  10$:    RSB                             ; Done
```

```
                                2C72  7730          .SBTTL  NET$ADJ_LPD_CRI - Locate CNF given ADJ index
                                2C72  7731 ;+
                                2C72  7732 ; NET$ADJ_LPD_CRI - Locate CNR and CNF given ADJ index
                                2C72  7733 ;
                                2C72  7734 ; INPUTS         R11-R9  Scratch
                                2C72  7735 ;                R8      Low byte contains ADJ index
                                2C72  7736 ;                R7,R6   Scratch
                                2C72  7737 ;
                                2C72  7738 ; OUTPUTS:       R11     CNR address
                                2C72  7739 ;                R10     CNF address
                                2C72  7740 ;                R9-R8   Garbage
                                2C72  7741 ;                R7      ADJ address
                                2C72  7742 ;                R6      LPD address
                                2C72  7743 ;                R0      Low bit set if successful
                                2C72  7744 ;                        Low bit clear otherwise
                                2C72  7745 ;-
                                2C72  7746 NET$ADJ_LPD_CRI::
 5B   00000000'EF    D0         2C72  7747          MOVL    NET$GL_CNR_CRI,R11      ; Get data base root for CRI
                5A   D4         2C79  7748          CLRL    R10                     ; No CNF yet
                49   10         2C7B  7749          BSBB    NET$FIND_ADJ            ; Find LPD & ADJ via index in R8
              1B 50  E9         2C7D  7750          BLBC    R0,10$                  ; If LPD then none
           50    5B   D0         2C80  7751          MOVL    R11,R0                  ; Make a copy
        58    20 A6   3C         2C83  7752          MOVZWL  LPD$W_PTH(R6),R8        ; Get full LPD path i.d.
           50    60   D0         2C87  7753 5$:      MOVL    CNF$L_FLINK(R0),R0      ; Get next CNF
              5B 50   D1         2C8A  7754          CMPL    R0,R11                  ; At head of list?
                 0C   13         2C8D  7755          BEQL    10$                     ; If EQL yes, return with LBC in R0
        12 A0    58   B1         2C8F  7756          CMPW    R8,CNF$W_ID(R0)         ; This it?
                 F2   12         2C93  7757          BNEQ    5$                      ; If NEQ keep trying
              5A 50   D0         2C95  7758          MOVL    R0,R10                  ; Copy CNF address
              50 00'  D0         2C98  7759          MOVL    S^#SS$_NORMAL,R0        ; Set status
                      05         2C9B  7760 10$:     RSB                             ; Done
```

```
                     2C9C  7762          .SBTTL  NET$LOCATE_LPD - Locate LPD given CNF
                     2C9C  7763 ;+
                     2C9C  7764 ; NET$LOCATE_LPD
                     2C9C  7765 ;
                     2C9C  7766 ; INPUTS:        R11      CNR address
                     2C9C  7767 ;                R10      CNF address
                     2C9C  7768 ;                R9-R6    Scratch
                     2C9C  7769 ;
                     2C9C  7770 ; OUTPUTS:       R11,R10  Preserved
                     2C9C  7771 ;                R9-R7    Garbage
                     2C9C  7772 ;                R6       LPD if low bit set in R0
                     2C9C  7773 ;                         Zero if low bit clear in R0
                     2C9C  7774 ;                R0       SS$_NORMAL     if successful
                     2C9C  7775 ;                         SS$_DEVINACT   otherwise
                     2C9C  7776 ;-
                     2C9C  7777 NET$LOCATE_LPD::
58   12 AA   3C      2C9C  7778          MOVZWL  CNF$W_ID(R10),R8        ; Get LPD i.d.
                     2CA0  7779                                          ; And fall thru
```

```
                                2CA0  7781          .SBTTL  NET$FIND_LPD - Find LPD given LPD index
                                2CA0  7782 ;+
                                2CA0  7783 ; NET$FIND_LPD - Find LPD given LPD index
                                2CA0  7784 ;
                                2CA0  7785 ; INPUTS:        R8      Low byte contains LPD index
                                2CA0  7786 ;                R6      Scratch
                                2CA0  7787 ;
                                2CA0  7788 ; OUTPUTS:       R8      Garbage
                                2CA0  7789 ;                R6      LPD if low bit set in R0
                                2CA0  7790 ;                        Zero if low bit clear in R0
                                2CA0  7791 ;                R0      SS$_NORMAL    if successful
                                2CA0  7792 ;                        SS$_DEVINACT  otherwise
                                2CA0  7793 ;-
                                2CA0  7794 NET$FIND_LPD::
     50   00000000'EF    D0     2CA0  7795          MOVL    NET$GL_PTR_VCB,R0       ; Get the RCB address
          58    58      9A     2CA7  7796          MOVZBL  R8,R8                   ; Get low byte of LPD index
                12      13     2CAA  7797          BEQL    10$                     ; If EQL then there's none
        5C A0   58      91     2CAC  7798          CMPB    R8,RCB$B_MAX_LPD(R0)    ; Within range ?
                0C      14     2CB0  7799          BGTR    10$                     ; If not, branch
     56    28 B048    D0     2CB2  7800          MOVL    @RCB$L_PTR_LPD(R0)[R8],R6 ; Get LPD address
                05      18     2CB7  7801          BGEQ    10$                     ; Branch if not valid
          50   00'     D0     2CB9  7802          MOVL    S^#SS$_NORMAL,R0        ; Indicate success
                07      11     2CBC  7803          BRB     15$                     ; Take common exit
                56      D4     2CBE  7804 10$:     CLRL    R6                      ; Nullify LPD pointer
     50   0000'8F    3C     2CC0  7805          MOVZWL  #SS$_DEVINACT,R0        ; Indicate failure
                05      05     2CC5  7806 15$:     RSB
```

```
                                2CC6          .SBTTL  NET$FIND_ADJ - Find LPD & ADJ given ADJ index
                                2CC6  ;+
                                2CC6  ; NET$FIND_ADJ - Find LPD & ADJ given ADJ index
                                2CC6  ;
                                2CC6  ; INPUTS:       R8      Low word contains ADJ index
                                2CC6  ;               R6-R7   Scratch
                                2CC6  ;
                                2CC6  ; OUTPUTS:      R8      Garbage
                                2CC6  ;               R7      ADJ address
                                2CC6  ;               R6      LPD address
                                2CC6  ;               R0      SS$_NORMAL    if successful
                                2CC6  ;                       SS$_DEVINACT  otherwise
                                2CC6  ;-
                                2CC6  NET$FIND_ADJ::
50   00000000'EF      D0       2CC6          MOVL    NET$GL_PTR_VCB,R0          ; Get the RCB address
          58    58    3C       2CCD          MOVZWL  R8,R8                      ; Get low word of ADJ index
                1C    13       2CD0          BEQL    10$                        ; If EQL then there's none
       68 A0    58    B1       2CD2          CMPW    R8,RCB$W_MAX_ADJ(R0)       ; Within range ?
                16    14       2CD6          BGTR    10$                        ; If not, branch
    57    2C B048    D0       2CD8          MOVL    @RCB$L_PTR_ADJ(R0)[R8],R7 ; Get ADJ address
      0D 67    00    E1       2CDD          BBC     #ADJ$V_INUSE,ADJ$B_STS(R7),10$ ; Branch if slot not in use
      58    02 A7    9A       2CE1          MOVZBL  ADJ$B_LPD_INX(R7),R8       ; Get LPD index
    56    28 B048    D0       2CE5          MOVL    @RCB$L_PTR_LPD(R0)[R8],R6 ; Get LPD address
          50    00'  D0       2CEA          MOVL    S^#SS$_NORMAL,R0           ; Indicate success
                     05       2CED          RSB
                                2CEE
                57    D4       2CEE  10$:    CLRL    R7                         ; Nullify pointer
    50    0000'8F    3C       2CF0          MOVZWL  #SS$_DEVINACT,R0           ; Indicate failure
                     05       2CF5          RSB
```

```
                                2CF6          .SBTTL  NET$GET_PLVECLPD - Find next active LPD
                                2CF6  ;+
                                2CF6  ; NET$GET_PLVECLPD      - Find next active LPD using the indicated line (PLVEC)
                                2CF6  ;
                                2CF6  ; INPUTS:       R4      PLVEC index
                                2CF6  ;               R1      Previous LPD index (scan starts with R1 +1)
                                2CF6  ;
                                2CF6  ; OUPUTS:       R1      New LPD address
                                2CF6  ;               R0      SS$_NORMAL    if successful
                                2CF6  ;                       SS$_DEVINACT  otherwise
                                2CF6  ;-
                                2CF6  NET$GET_PLVECLPD::                      ; Find next active LPD using this line
                  53    DD      2CF6          PUSHL   R3                      ; Save reg
53    00000000'EF     D0      2CF8          MOVL    NET$GL_PTR_VCB,R3       ; Get the RCB address
                  51    D6      2CFF  20$:    INCL    R1                      ; Start at next LPD
        5C A3     51    91      2D01          CMPB    R1,RCB$B_MAX_LPD(R3)    ; Within range ?
                  1A    1A      2D05          BGTRU   100$                    ; If not, branch
      50    28 B341   D0      2D07          MOVL    @RCB$L_PTR_LPD(R3)[R1],R0 ; Get next LPD address
                  F1    18      2D0C          BGEQ    20$                     ; Branch if not valid
                  00    E1      2D0E          BBC     #LPD$V_ACTIVE,-         ; Is LPD active ?
           EC 22 A0             2D10                      LPD$W_STS(R0),20$   ; If BC then no
        28 A0     54    91      2D13          CMPB    R4,LPD$B_PLVEC(R0)      ; Is it using the indicated line?
                  E6    12      2D17          BNEQ    20$                     ; If EQL yes, we've found the LPD
            51    50    D0      2D19          MOVL    R0,R1                   ; Copy LPD pointer
            50    00'   D0      2D1C          MOVL    S^#SS$_NORMAL,R0        ; Indicate success
                  07    11      2D1F          BRB     110$                    ; Take common exit
                  51    D4      2D21  100$:   CLRL    R1                      ; Nullify LPD pointer
      50    0000'8F     3C      2D23          MOVZWL  #SS$_DEVINACT,R0        ; Indicate failure
                  53 8ED0       2D28  110$:   POPL    R3                      ; Restore reg
                     05         2D2B          RSB
                                2D2C
```

```
                                2D2C          .SBTTL  TELL_NETDRIVER - Inform NETDRIVER of an event
                                2D2C  ;+
                                2D2C  ; TELL_NETDRIVER        - Inform NETDRIVER of an event
                                2D2C  ;
                                2D2C  ; INPUTS:       R0 = Function code (NETUPD$_DLL_)
                                2D2C  ;               R6 = LPD address
                                2D2C  ;
                                2D2C  ; OUTPUTS:      R0 = Status
                                2D2C  ;
                                2D2C  ;               All other registers are preserved.
                                2D2C  ;-
                                2D2C  TELL_NETDRIVER:                                ;
                     3E    BB   2D2C          PUSHR   #^M<R1,R2,R3,R4,R5>          ; Save critical regs
      55   00000000'EF    D0   2D2E          MOVL    NET$GL_NET_UCB,R5            ; Get the ACP's NET UCB
      52   00000000'EF    D0   2D35          MOVL    NET$GL_PTR_VCB,R2            ; Get RCB
                51    56    D0   2D3C          MOVL    R6,R1                        ; Get the LPD address
                   D2BE'    30   2D3F          BSBW    CALL_NETDRIVER               ; Tell NETDRIVER
                     3E    BA   2D42          POPR    #^M<R1,R2,R3,R4,R5>          ; Restore regs
                           05   2D44          RSB
                                2D45  .END
```

```
$$T1                    =  00000000
$$_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
ACT_ADJ_DOWN              00001FA5 R      06
ACT_BC_UP                 00001D83 R      06
ACT_BUG                   00000E8E R      06
ACT_DLL_UP                00001AF0 R      06
ACT_ELECT                 000011EB R      06
ACT_ENT_DLE               00001B9A R      06
ACT_ENT_MOP               00001AE6 R      06
ACT_ENT_MPR               00001F90 R      06
ACT_ENT_RUN               00001D0A R      06
ACT_EXIT                  00000E96 R      06
ACT_EXI_SERV              00001BCA R      06
ACT_FAILED                00001C25 R      06
ACT_INI_FAIL              00001BFE R      06
ACT_LOG_ADE               00000EAE R      06
ACT_LOG_CDE               00000EA4 R      06
ACT_LOG_NFE               00000EB8 R      06
ACT_NOP                   00000E9D R      06
ACT_NYI                   00000E92 R      06
ACT_PVC_START             0000251F R      06
ACT_QIO_SHUT              0000227B R      06
ACT_QIO_STRT              000022F0 R      06
ACT_RCV_2STR              00000EC2 R      06
ACT_RCV_ART               00001373 R      06
ACT_RCV_ARTA              00001382 R      06
ACT_RCV_EHEL              000011FE R      06
ACT_RCV_RHEL              000010D9 R      06
ACT_RCV_RT                0000124F R      06
ACT_RCV_RTA               0000125E R      06
ACT_RCV_STR               00000EC6 R      06
ACT_RCV_VRF               00000FFB R      06
ACT_REQ_UPDATE            00001424 R      06
ACT_RUN_DOWN              00001C30 R      06
ACT_RUN_SHUT              00001F9C R      06
ACT_RUN_SYNC              00001F74 R      06
ACT_RUN_UXPK              00001F82 R      06
ACT_SET_OPER              00001C83 R      06
ACT_SYN_FAIL              00001BEA R      06
ACT_TST_DL                00001CA6 R      06
ACT_X25_CALL              000025BE R      06
ACT_X25_RESET             00001C0D R      06
ACT_XMT                   0000268E R      06
ADAPT_TO_PARTNER          00000FAD R      06
ADJ                       00000786 R      06
ADJ$B_BCPRI             =  0000000C
ADJ$B_LPD_INX           =  00000002
ADJ$B_PTYPE             =  00000001
ADJ$B_STS               =  00000000
ADJ$C_LENGTH                =  0000000D
ADJ$C_PTY_AREA              =  00000003
ADJ$C_PTY_PH2               =  00000002
ADJ$C_PTY_PH3               =  00000000
ADJ$C_PTY_PH3N              =  00000001
ADJ$C_PTY_PH4               =  00000004
ADJ$C_PTY_PH4N              =  00000005
ADJ$C_PTY_UNK               =  FFFFFFFF
ADJ$M_INUSE                 =  00000001
ADJ$M_LSN                   =  00000008
ADJ$M_RTG                   =  00000004
ADJ$M_RUN                   =  00000002
ADJ$V_INUSE                 =  00000000
ADJ$V_LSN                   =  00000003
ADJ$V_RTG                   =  00000002
ADJ$V_RUN                   =  00000001
ADJ$W_BUFSIZ                =  00000006
ADJ$W_INT_LSN               =  00000008
ADJ$W_LPD                   =  00000002
ADJ$W_PNA                   =  00000004
ADJ$W_TIM_LSN               =  0000000A
ADJ_DOWN                     000020C5 R      06
ADJ_DOWN_EVENT               00000862 R      06
ALLOC_COSTHOPS               000003C1 R      06
ALLOC_LPD                    00000168 R      06
APL                          00000874 R      06
AREA_DECISION                00001801 R      06
BEA_UP                       00001EE0 R      06
BIT...                      =  0000000A
BRA_DOWN                     00002119 R      06
BRA_UP                       00001DE9 R      06
BUG$_NETNOSTATE              ********   X    06
BUILD_RTR_LIST               0000219D R      06
CALL_NETDRIVER               ********   X    06
CCB$C_UCB                   =  00000000
CHECK_REQ_PARAMS             000004DE R      06
CHK_CIRC_START               000025F4 R      06
CHK_IO                       00002AB3 R      06
CHK_RUS4                     00000C12 R      06
CNF$GET_FIELD                ********   X    06
CNF$KEY_SEARCH               ********   X    06
CNF$L_FLINK                 =  00000000
CNF$PUT_FIELD                ********   X    06
CNF$W_ID                    =  00000012
CNF$_ADVANCE                =  00000000
CNF$_QUIT                   =  00000002
CNF$_TAKE_CURR              =  00000003
CNF$_TAKE_PREV              =  00000001
COND_DEAL_LPD                00000424 R      06
CRC16                        00000106 R      02
CRD                          00000720 R      06
CXB$C_OVERHEAD              =  0000004C
DDT$L_FDT                   =  00000008
DEAL_LPD                     0000044D R      06
DECISION                     0000158B R      06
DEVTRN$C_DEV_CI             =  00000004
DEVTRN$C_DEV_DMC            =  00000001
```

```
DEVTRN$C_DEV_PPUNA              =  0000000A
DEVTRN$C_DEV_UNA                =  00000009
DISPATCH                           000006FC  R      06
DLE$BC_DOWN                        ********    X   06
DLE$BC_UP                          ********    X   06
DLE$LPD_STATUS                     ********    X   06
DLE$MOP_REQUEST                    ********    X   06
ELECT_ROUTER                       00002220  R      06
ENDNODE_DECISION                   00001A96  R      06
EVC$C_TPL_ACH                   =  00000111
EVC$C_TPL_APL                   =  00000100
EVC$C_TPL_ARJ                   =  00000110
EVC$C_TPL_AUP                   =  0000010F
EVC$C_TPL_IOF                   =  0000010D
EVC$C_TPL_ISF                   =  0000010C
EVC$C_TPL_LDF                   =  00000107
EVC$C_TPL_LDO                   =  00000113
EVC$C_TPL_LDS                   =  00000112
EVC$C_TPL_LUP                   =  0000010A
EVC$C_TPL_OPL                   =  00000103
EVC$C_TPL_PFM                   =  00000104
EVC$C_TPL_PRSN_ADJB             =  00000008
EVC$C_TPL_PRSN_ADJC             =  00000004
EVC$C_TPL_PRSN_ADJR             =  00000007
EVC$C_TPL_PRSN_DROP             =  0000000E
EVC$C_TPL_PRSN_LTMO             =  0000000A
EVC$C_TPL_PRSN_RUCS             =  00000003
EVC$C_TPL_PRSN_SYNC             =  00000000
EVC$C_TPL_PRSN_UXPK             =  00000002
EVC$C_TPL_PRSN_VREQ             =  0000000D
EVC$C_TPL_PRSN_VRSK             =  00000006
EVC$C_TPL_PRU                   =  00000105
EVC$C_TPL_PSTS_RCH              =  00000000
EVC$C_TPL_PSTS_URC              =  00000001
EVC$C_TPL_RCH                   =  0000010E
EVC$C_TPL_RPL                   =  00000102
EVC$C_TPL_UPL                   =  00000101
EVC$C_TPL_VFR                   =  00000106
EXE$GL_ABSTIM                      ********    X   06
EXIT_RUN_STATE                     00001FC8  R      06
FDT_IOTYPE                      =  00000008
FDT_LEGAL                       =  00000000
FILL_JNL                           00000E15  R      06
FIND_ENDNODE_BEA                   000017CE  R      06
FIND_PATH_TO_AREA                  000018BD  R      06
FIND_PATH_TO_NODE                  00001700  R      06
FIND_WQE_CTX                       00000E4A  R      06
FORCE_FULL_DECISION                0000002A  R      06
FUNC                            =  0000001C
IGNORE_MSG                         000007F9  R      06
IO$M_ABORT                         ********    X   06
IO$M_ACCEPT                        ********    X   06
IO$M_SHUTDOWN                      ********    X   06
IO$M_STARTUP                       ********    X   06
IO$_ACCESS                         ********    X   06
IO$_DEACCESS                       ********    X   06
IO$_NETCONTROL                     ********    X   06
IO$_READLBLK                       ********    X   06
IO$_SETMODE                        ********    X   06
IO$_WRITELBLK                      ********    X   06
IOC$VERIFYCHAN                     ********    X   06
IOSB                                            =  00000000
IOWQE_LENGTH                                    =  00000020
IRP                                                00000073E R      06
IRP$L_IOST1                                     =  00000038
IRP$L_IOST2                                     =  0000003C
IRP$L_SVAPTE                                    =  0000002C
IRP$W_FUNC                                      =  00000020
KILL_WQE                                           00000D88  R      06
LEV$AW_STA_TAB                                     00000000  R      03
LEV$C_ADJ_DOWN                                  =  00000012    G
LEV$C_BC_UP                                     =  00000013    G
LEV$C_BUG                                       =  00000002    G
LEV$C_DLE_ACC                                   =  0000001E    G
LEV$C_ELECT_TIM                                 =  0000001B    G
LEV$C_ENT_DCE                                   =  0000001D    G
LEV$C_EVENTS                                    =  00000025
LEV$C_EXIT                                      =  00000001    G
LEV$C_FAILED                                    =  0000001C    G
LEV$C_IO_FAIL                                   =  00000015    G
LEV$C_IO_SUCC                                   =  00000016    G
LEV$C_IO_TIMOUT                                 =  00000014    G
LEV$C_IRP_DOWN                                  =  00000020    G
LEV$C_IRP_MM                                    =  00000021    G
LEV$C_IRP_RESET                                 =  0000001F    G
LEV$C_LIN_DOWN                                  =  00000011    G
LEV$C_LIN_UP                                    =  00000010    G
LEV$C_LOG_ADE                                   =  00000024    G
LEV$C_LOG_CDE                                   =  00000023    G
LEV$C_LOG_NFE                                   =  00000022    G
LEV$C_MAX_EVT                                   =  00000024
LEV$C_NO_EVT                                    =  00000000    G
LEV$C_OPR_OFF                                   =  00000005    G
LEV$C_OPR_ON                                    =  00000006    G
LEV$C_OPR_SRV                                   =  00000007    G
LEV$C_PVC_START                                 =  00000018    G
LEV$C_RCV_ART                                   =  0000000C    G
LEV$C_RCV_EHEL                                  =  0000000E    G
LEV$C_RCV_RHEL                                  =  0000000D    G
LEV$C_RCV_RT                                    =  0000000B    G
LEV$C_RCV_STR                                   =  00000008    G
LEV$C_RCV_VRF                                   =  00000009    G
LEV$C_RCV_VVF                                   =  0000000A    G
LEV$C_REQ_SHUT                                  =  00000004    G
LEV$C_STATES                                    =  00000010
LEV$C_STA_.                                     =  00000000
LEV$C_STA_A                                     =  00000004
LEV$C_STA_B                                     =  00000005
LEV$C_STA_C                                     =  00000006
LEV$C_STA_D                                     =  00000007
LEV$C_STA_J                                     =  00000008
LEV$C_STA_M                                     =  00000003
LEV$C_STA_R                                     =  00000009
LEV$C_STA_S                                     =  00000000
```

```
LEV$C_STA_W                     =  00000001
LEV$C_STA_Y                     =  00000002
LEV$C_STRT_TIM                  =  0000001A  G
LEV$C_UNJAM                     =  00000003  G
LEV$C_X25_CALL                  =  00000017  G
LEV$C_X25_RESET                 =  00000019  G
LEV$C_XMT_IDLE                  =  0000000F  G
LEV_AC_ACTTAB                      00000000 R      02
LEV_B_PRIORITY                     0000001C R      04
LEV_L_ADJ                          00000010 R      04
LEV_L_LPD                          0000000C R      04
LEV_Q_CRI                          00000004 R      04
LEV_Q_PSWDESC                      00000024 R      04
LEV_W_BLKSIZE                      00000013 R      04
LEV_W_HELLO                        00000020 R      04
LEV_W_PNA                          00000014 R      04
LOWEST_PRIO_BRA                    00001EAD R      06
LPD$B_ASRM_LEFT                 =  00000055
LPD$B_ASRM_POS                  =  00000054
LPD$B_ASTCNT                    =  0000001B
LPD$B_BCPRI                     =  0000002A
LPD$B_CNT_IFL                   =  0000004F
LPD$B_CNT_LDN                   =  0000004E
LPD$B_COST                      =  00000029
LPD$B_ETY                       =  0000001D
LPD$B_IRPCNT                    =  0000001C
LPD$B_PLVEC                     =  00000028
LPD$B_PTH_INX                   =  00000020
LPD$B_PVCFLG                    =  00000025
LPD$B_SRM_LEFT                  =  00000053
LPD$B_SRM_POS                   =  00000052
LPD$B_STARTUPS                  =  0000000B
LPD$B_STI                       =  00000026
LPD$B_SUB_STA                   =  00000027
LPD$B_TSTCNT                    =  0000001A
LPD$B_XMTFLG                    =  00000024
LPD$B_XMT_IPL                   =  0000001F
LPD$B_XMT_SRL                   =  0000001E
LPD$C_ASRM_AREAS                =  00000040
LPD$C_ASRM_SHFT                 =  00000006
LPD$C_ASRM_SIZE                 =  00000001
LPD$C_LENGTH                    =  0000006A
LPD$C_LOC_INX                   =  00000001
LPD$C_SRM_NODES                 =  00000020
LPD$C_SRM_SHFT                  =  00000005
LPD$C_SRM_SIZE                  =  00000020
LPD$G_ASRM                      =  0000005E
LPD$G_SRM                       =  00000056
LPD$G_XMT_ASRM                  =  00000062
LPD$G_XMT_SRM                   =  0000005A
LPD$L_ABS_TIM                   =  00000036
LPD$L_RCV_IRP                   =  00000032
LPD$L_RTR_LIST                  =  0000002E
LPD$L_UCB                       =  00000010
LPD$M_PVC_ACCESS                =  00000001
LPD$M_PVC_RESET                 =  00000004
LPD$M_PVC_RESTRT                =  00000002
LPD$M_TOGGLE                    =  00001000
LPD$M_XMT_IDLE                  =  00000008
LPD$M_XMT_STR                   =  00000002
LPD$M_XMT_VRF                   =  00000004
LPD$Q_REQ_WAIT                  =  00000000
LPD$V_ACCESS                    =  00000003
LPD$V_ACTIVE                    =  00000000
LPD$V_ALIGNQ                    =  0000000E
LPD$V_ALIGNW                    =  0000000D
LPD$V_BC                        =  0000000A
LPD$V_DLE                       =  00000002
LPD$V_ELECT_TIM                 =  0000000F
LPD$V_INCOMING                  =  00000009
LPD$V_PVC_ACCESS                =  00000000
LPD$V_PVC_ACCESSED              =  00000007
LPD$V_PVC_RESET                 =  00000002
LPD$V_PVC_RESTRT                =  00000001
LPD$V_RBF                       =  00000006
LPD$V_RUN                       =  00000004
LPD$V_STRTIM                    =  00000001
LPD$V_TOGGLE                    =  0000000C
LPD$V_X25                       =  00000007
LPD$V_XBF                       =  00000005
LPD$V_XMT_ART                   =  00000006
LPD$V_XMT_DALLY                 =  00000000
LPD$V_XMT_IDLE                  =  00000003
LPD$V_XMT_RT                    =  00000004
LPD$V_XMT_STR                   =  00000001
LPD$V_XMT_VRF                   =  00000002
LPD$W_BUFSIZ                    =  00000050
LPD$W_CHAN                      =  00000014
LPD$W_DRT                       =  0000002C
LPD$W_INT_TLK                   =  00000018
LPD$W_PTH                       =  00000020
LPD$W_STS                       =  00000022
LPD$W_TIM_TLK                   =  00000016
LSN                                0000082E R      06
MAX_COST                           00000030 R      04
MAX_HOPS                           0000002C R      04
MAX_SRL                            00000146 R      02
MC1                                000027F5 R      06
MOVIT                              000027ED R      06
MOVITU                             000027F8 R      06
MSG_MAP_TABLE                      000000B4 R      02
NET$AB_EVT_WQE                     ******** *    X 06
NET$ADJ_LPD_CRI                    00002C72 RG     06
NET$ALLOCATE                       ******** *    X 06
NET$ALONPGD_Z                      ******** *    X 06
NET$AL_AREA_CH                     00001A88 RG     05
NET$AL_CH_VEC                      00000980 RG     05
NET$AW_AREA_C_H                    00000900 RG     05
NET$AW_MIN_C_A                     00000100 RG     05
NET$C_ACT_TIMER                 =  0000001E
NET$C_EFN_ASYN                  =  00000002
NET$C_EFN_WAIT                  =  00000001
NET$C_IPL                       =  00000008
NET$C_MAXACCFLD                 =  00000027
```

```
NET$C_MAXLINNAM         = 0000000F
NET$C_MAXLNK            = 000003FF
NET$C_MAXNODNAM         = 00000006
NET$C_MAXOBJNAM         = 0000000C
NET$C_MAX_AREAS         = 0000003F
NET$C_MAX_LINES         = 00000040
NET$C_MAX_NCB           = 0000006E
NET$C_MAX_NODES         = 000003FF
NET$C_MAX_OBJ           = 000000FF
NET$C_MAX_WQE           = 00000014
NET$C_MINBUFSIZ         = 000000C0
NET$C_TID_ACT           = 00000003
NET$C_TID_RUS           = 00000001
NET$C_TID_XRT           = 00000002
NET$C_TRCTL_CEL         = 00000002
NET$C_TRCTL_OVR         = 00000005
NET$C_UTLBUFSIZ         = 00001000
NET$DEALLOCATE            ********    X   06
NET$DLLQIOAST             00002BD6 R      06
NET$DLLUPDLNI             00000000 RG     07
NET$DLL_ALL_OFF           0000003C RG     06
NET$DLL_OPR_SET           00000071 RG     06
NET$DLL_PRC_WQE           00000D80 R      06
NET$DLL_QIO_CO            00002AD1 R      06
NET$DLL_RCV               00000671 RG     06
NET$DLL_X25_CALL          00000577 RG     06
NET$DLL_X25_RESET         00000630 RG     06
NET$EVT_INTRAW            ********    X   06
NET$FIND_ADJ              00002CC6 RG     06
NET$FIND_LPD              00002CA0 RG     06
NET$GET_LPD_CRI           00002C48 RG     06
NET$GET_PLVECLPD          00002CF6 RG     06
NET$GET_RTG               ********    X   06
NET$GET_RTG2              ********    X   06
NET$GET_RTG3              ********    X   06
NET$GET_VEC               ********    X   06
NET$GET_VEC2              ********    X   06
NET$GET_VEC3              ********    X   06
NET$GL_CNR_CRI            ********    X   06
NET$GL_CNR_LNI            ********    X   06
NET$GL_CNR_NDI            ********    X   06
NET$GL_CNR_PLI            ********    X   06
NET$GL_INITVER            00000000 RG     04
NET$GL_NET_UCB            ********    X   06
NET$GL_PTR_LNI            ********    X   06
NET$GL_PTR_VCB            ********    X   07
NET$GQ_MBX_NAME           ********    X   06
NET$GW_X25_CHAN           ********    X   06
NET$G_ALL_ROU             00000100 R      02
NET$INIT_ROUTING          00000000 RG     06
NET$JNX_CO                ********    X   06
NET$LOCATE_LPD            00002C9C RG     06
NET$LOCATE_NDI            ********    X   06
NET$LOCLPD_DOWN           ********    X   06
NET$M_MAXLNKMSK         = 000003FF
NET$NDI_BY_ADD            ********    X   06
NET$SET_QIOW              ********    X   06

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$_DLL_ON                  = 00000005
NETUPD$_REACT_RCV               = 0000000C
NETUPD$_SEND_HELLO              = 0000000D
NFB$C_CRI_BLK                   = 04000003
NFB$C_CRI_CHR                   = 04020043
NFB$C_CRI_COS                   = 04010018
NFB$C_CRI_HET                   = 04010019
NFB$C_CRI_MBL                   = 04010022
NFB$C_CRI_MRC                   = 0401001B
NFB$C_CRI_MWI                   = 04010023
NFB$C_CRI_NAM                   = 04020041
NFB$C_CRI_NUM                   = 04020048
NFB$C_CRI_RCT                   = 0401001C
NFB$C_CRI_RPR                   = 04010035
NFB$C_CRI_STA                   = 04010013
NFB$C_CRI_TYP                   = 04010020
NFB$C_CRI_USE                   = 0401001F
NFB$C_CRI_VER                   = 04000004
NFB$C_CRI_VMSNAM                = 04020042
NFB$C_CRI_XPT                   = 04010033
NFB$C_LNI_AMC                   = 01010030
NFB$C_LNI_AMH                   = 01010031
NFB$C_LNI_BRT                   = 0101002C
NFB$C_LNI_IDE                   = 01020043
NFB$C_LNI_MCO                   = 01010020
NFB$C_LNI_MHO                   = 01010021
NFB$C_LNI_NAM                   = 01020041
NFB$C_LNI_RSI                   = 0101001C
NFB$C_LNI_RTI                   = 0101001B
NFB$C_NDI_RPA                   = 02020054
NFB$C_NDI_TPA                   = 02020055
NFB$C_OP_EQL                    = 00000000
NFB$C_PLI_BFN                   = 0501001E
NFB$C_PLI_BUS                   = 0501001F
NFB$C_PLI_CHR                   = 05020043
NFB$C_PLI_DEVNAM                = 05020047
NFB$C_PLI_PLVEC                 = 05010020
NFB$C_PLI_VMSNAM                = 05020042
NMA$C_CIRTY_X25                 = 00000003
NMA$C_CIRUS_INC                 = 00000001
NMA$C_CIRUS_OUT                 = 00000002
NMA$C_CIRUS_PER                 = 00000000
NMA$C_CIRVE_DIS                 = 00000001
NMA$C_CIRVE_ENA                 = 00000000
NMA$C_CIRXPT_NR4                = 00000004
NMA$C_CIRXPT_PH2                = 00000002
NMA$C_CIRXPT_PH3                = 00000003
```

```
NMA$C_LINSS_ASE                 = 00000006
NMA$C_LINSS_FAI                 = 0000000B
NMA$C_LINSS_STA                 = 00000000
NMA$C_LINSS_SYN                 = 0000000A
NMA$C_STATE_OFF                 = 00000001
NMA$C_STATE_ON                  = 00000000
NMA$C_STATE_SER                 = 00000002
NOL                               0000089E  R      06
NON_FATAL                         000008B1  R      06
NSP$$$_QUAL_ACK                 = 00000000
NSP$$$_QUAL_ALTFLW              = 00000000
NSP$$$_QUAL_DATA                = 00000000
NSP$$$_QUAL_FLW                 = 00000000
NSP$$$_QUAL_INF                 = 00000000
NSP$$$_QUAL_MSG                 = 00000000
NSP$$$_QUAL_SRV                 = 00000000
NSP$C_EXT_LNK                   = 0000001E
NSP$C_FLW_DATA                  = 00000000
NSP$C_FLW_INT                   = 00000001
NSP$C_FLW_NOP                   = 00000000
NSP$C_FLW_XOFF                  = 00000001
NSP$C_FLW_XON                   = 00000002
NSP$C_HSZ_ACK                   = 00000007
NSP$C_HSZ_CA                    = 00000003
NSP$C_HSZ_CC                    = 00000064
NSP$C_HSZ_CD                    = 000000F0
NSP$C_HSZ_CI                    = 000000F0
NSP$C_HSZ_DATA                  = 00000009
NSP$C_HSZ_DC                    = 00000016
NSP$C_HSZ_DI                    = 00000016
NSP$C_HSZ_INT                   = 00000009
NSP$C_HSZ_LS                    = 00000009
NSP$C_INF_V31                   = 00000001
NSP$C_INF_V32                   = 00000000
NSP$C_INF_V33                   = 00000002
NSP$C_MAXADR                    = 00000009
NSP$C_MSG_CA                    = 00000024
NSP$C_MSG_CC                    = 00000028
NSP$C_MSG_CI                    = 00000018
NSP$C_MSG_DATA                  = 00000000
NSP$C_MSG_DC                    = 00000048
NSP$C_MSG_DI                    = 00000038
NSP$C_MSG_DTACK                 = 00000004
NSP$C_MSG_INT                   = 00000030
NSP$C_MSG_LIACK                 = 00000014
NSP$C_MSG_LS                    = 00000010
NSP$C_SRV_MFC                   = 00000002
NSP$C_SRV_NFC                   = 00000000
NSP$C_SRV_REQ                   = 00000001
NSP$C_SRV_SFC                   = 00000001
NSP$M_ACK_NAK                   = 00001000
NSP$M_ACK_NUM                   = 00000FFF
NSP$M_ACK_VALID                 = 00008000
NSP$M_DATA_BOM                  = 00000020
NSP$M_DATA_EOM                  = 00000040
NSP$M_DATA_OVFW                 = 00000080
NSP$M_FLW_CHAN                  = 0000000C
NSP$M_FLW_DRV                   = 000000F0
NSP$M_FLW_INT                   = 00000020
NSP$M_FLW_INUSE                 = 00000010
NSP$M_FLW_LISUB                 = 00000004
NSP$M_FLW_MODE                  = 00000003
NSP$M_FLW_SP1                   = 00000008
NSP$M_FLW_SP2                   = 00000040
NSP$M_FLW_SP3                   = 00000080
NSP$M_FLW_XOFF                  = 00000001
NSP$M_FLW_XON                   = 00000002
NSP$M_INF_VER                   = 00000003
NSP$M_MSG_INT                   = 00000020
NSP$M_MSG_LI                    = 00000010
NSP$M_SRV_01                    = 00000003
NSP$M_SRV_EXT                   = 00000080
NSP$M_SRV_FLW                   = 0000000C
NSP$M_SRV_REQ                   = 000000F3
NSP$M_SRV_SP1                   = 00000070
NSP$R_QUAL                      = 00000000
NSP$S_ACK_NUM                   = 0000000C
NSP$S_ACK_SP2                   = 00000002
NSP$S_DATA_SP                   = 00000005
NSP$S_FLW_CHAN                  = 00000002
NSP$S_FLW_DRV                   = 00000004
NSP$S_FLW_MODE                  = 00000002
NSP$S_INF_VER                   = 00000002
NSP$S_MSG_SP1                   = 00000004
NSP$S_NSPMSG                    = 00000005
NSP$S_QUAL                      = 00000005
NSP$S_QUAL_ACK                  = 00000002
NSP$S_QUAL_ALTFLW               = 00000001
NSP$S_QUAL_DATA                 = 00000001
NSP$S_QUAL_FLW                  = 00000001
NSP$S_QUAL_INF                  = 00000001
NSP$S_QUAL_MSG                  = 00000005
NSP$S_QUAL_SRV                  = 00000001
NSP$S_SRV_01                    = 00000002
NSP$S_SRV_FLW                   = 00000002
NSP$S_SRV_SP1                   = 00000003
NSP$V_ACK_NAK                   = 0000000C
NSP$V_ACK_NUM                   = 00000000
NSP$V_ACK_SP2                   = 0000000D
NSP$V_ACK_VALID                 = 0000000F
NSP$V_DATA_BOM                  = 00000005
NSP$V_DATA_EOM                  = 00000006
NSP$V_DATA_OVFW                 = 00000007
NSP$V_DATA_SP                   = 00000000
NSP$V_FLW_CHAN                  = 00000002
NSP$V_FLW_DRV                   = 00000004
NSP$V_FLW_INT                   = 00000005
NSP$V_FLW_INUSE                 = 00000004
NSP$V_FLW_LISUB                 = 00000002
NSP$V_FLW_MODE                  = 00000000
NSP$V_FLW_SP1                   = 00000003
NSP$V_FLW_SP2                   = 00000006
NSP$V_FLW_SP3                   = 00000007
NSP$V_FLW_XOFF                  = 00000000
```

```
NSP$V_FLW_XON                   = 00000001
NSP$V_INF_VER                   = 00000000
NSP$V_MSG_INT                   = 00000005
NSP$V_MSG_LI                    = 00000004
NSP$V_MSG_SP1                   = 00000000
NSP$V_SRV_01                    = 00000000
NSP$V_SRV_EXT                   = 00000007
NSP$V_SRV_FLW                   = 00000002
NSP$V_SRV_SP1                   = 00000004
NSP$W_DSTLNK                    = 00000001
NSP$W_SRCLNK                    = 00000003
NUL                               00000889 R      06
NULL                              0000003C R      04
NUM_AREAS                       = 00000040
NUM_CIRCS                       = 00000041
NUM_NODES                       = 00000400
OPL                               0000086A R      06
OPR_EVT_MAP                       000000FC R      02
P1                              = 00000018
P2                              = 00000014
P3                              = 00000010
P4                              = 0000000C
P5                              = 00000008
PARSE_PH2_ADDR                    00000C8B R      06
PARSE_PH3_ADDR                    00000CB8 R      06
PARSE_PH4_ADDR                    00000CB8 R      06
PARSE_VERSION                     00000D04 R      06
PFE                               0000084F R      06
PLVEC$AB_DEV                      ********   X    06
PLVEC$AB_REFC                     ********   X    06
PLVEC$AB_STATE                    ********   X    06
PLVEC$AL_UCB                      ********   X    06
PLVEC$AW_CHAN                     ********   X    06
PLVEC$GB_MAX                      ********   X    06
PR$_IPL                           ********   X    07
PROC_ART                          0000138E R      06
PROC_EVT                          00000D93 R      06
PROC_RT                           0000126A R      06
PSI$C_NCB_PKTSIZE               = 00000015
PSI$C_NCB_PVCNAM                = 00000018
PSI$C_NCB_REMDTE                = 00000001
PSI$C_NCB_WINSIZE               = 00000016
PSI$C_RESET                     = 00000003
PSI$C_RESTART                   = 00000007
PTYPE                             00000038 R      04
PTY_TO_PHASE                      0000014A R      02
PTY_TO_VERSION                    00000154 R      02
QIOAST                            00002BE0 R      06
RCB$B_ACT_DLL                   = 00000060
RCB$B_CNT_APL                   = 00000095
RCB$B_CNT_NOL                   = 00000094
RCB$B_CNT_PFE                   = 00000097
RCB$B_CNT_RUL                   = 00000098
RCB$B_ETY                       = 0000008A
RCB$B_HOMEAREA                  = 0000008B
RCB$B_MAX_AREA                  = 0000008C
RCB$B_MAX_LPD                   = 0000005C
RCB$B_MAX_SNK                   = 0000005D
RCB$B_STATUS                    = 0000000B
RCB$B_STI                       = 00000061
RCB$L_PTR_ADJ                   = 0000002C
RCB$L_PTR_AOA                   = 00000020
RCB$L_PTR_JNX                   = 00000018
RCB$L_PTR_LPD                   = 00000028
RCB$L_PTR_OA                    = 0000001C
RCB$V_LVL2                      = 00000000
RCB$W_ADDR                      = 0000000E
RCB$W_ALIAS                     = 0000008D
RCB$W_CNT_NUL                   = 0000009A
RCB$W_DRT                       = 000000AA
RCB$W_ECLSEGSIZ                 = 0000007C
RCB$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_MCOUNT                    = 00000054
RCB$W_TOTBUFSIZ                 = 0000007E
RCV_ART                           00000BC3 R      06
RCV_EHEL                          00000A91 R      06
RCV_RHEL                          00000A0E R      06
RCV_RT                            00000AE8 R      06
RCV_RT3                           00000AF5 R      06
RCV_RT4                           00000B66 R      06
RCV_STR2                          000008B9 R      06
RCV_STR3                          00000915 R      06
RCV_STR4                          00000971 R      06
RCV_VRF2                          000009CF R      06
RCV_VRF3                          000009DF R      06
RCV_VRF4                          000009DF R      06
REACH_EVT                         00000000 R      05
REQUEST_UPDATE                    0000142D R      06
RESET_CHAN                        00002C39 R      06
RTGFLG                            00000040 R      04
RTG_CHG                           00000080 R      05
RTG_CHG_LEN                     = 00000080
RTG_V_ROS                       = 00000000
RTG_V_UPD                       = 00000001
SET_DLL_EVT                       00000D5B RG     06
SET_IOTIM                         00002C20 R      06
SIZ...                          = 00000001
SS$_BADPARAM                      ********   X    06
SS$_DEVINACT                      ********   X    06
SS$_INSFARG                       ********   X    06
SS$_INSFMEM                       ********   X    06
SS$_NORMAL                        ********   X    06
SS$_NOSUCHDEV                     ********   X    06
SS$_RESET                         ********   X    06
START_XRT                         00001A4A R      06
STR2                              00002727 R      06
STR3                              0000277A R      06
STR4                              000027AC R      06
STRT_TIMER_TICK                   00002412 R      06
```

```
SYS$ASSIGN                  ********  GX   06
SYS$CANCEL                  ********  GX   06
SYS$DASSGN                  ********  GX   06
SYS$QIO                     ********  GX   06
TELL_NETDRIVER              00002D2C  R    06
TIMER_RUS                   00001489  R    06
TIMER_XRT                   00001A29  R    06
TOGGLE_LINE                 00002633  R    06
TR$C_MAXHDR                 = 0000001C
TR$C_NI_ALLEND1             = 040000AB
TR$C_NI_ALLEND2             = 00000000
TR$C_NI_ALLROU1             = 030000AB
TR$C_NI_ALLROU2             = 00000000
TR$C_NI_PREFIX              = 000400AA
TR$C_NI_PROT                = 00000360
TR$C_PRI_ECL                = 0000001F
TR$C_PRI_RTHRU              = 0000001F
TR$C_TIM_DALLY              = 00000002
TR$C_TIM_DLLIO              = 000000B4
TR$C_TIM_DRDELAY            = 00000005
TR$C_TIM_RESTRT             = 0000000A
TR2C_INI_STR                = 00000001
TR2C_INI_VRF                = 00000002
TR2C_MAX_PNA                = 000000FF
TR2C_MSG_INI                = 00000058
TR2C_MSG_NOP                = 00000008
TR2C_NOP_LNG                = 00000001
TR2C_NUM_NOP                = 00000000
TR2C_PSW_LNG                = 00000008
TR2C_STR_FCT                = 00000000
TR2C_STR_LNG                = 0000000A
TR2C_STR_MXL                = 00000050
TR2C_STR_REQ                = 00000006
TR2C_VRF_LNG                = 00000002
TR2M_FCT_INT                = 00000002
TR2M_REQ_VRF                = 00000001
TR2V_REQ_VRF                = 00000000
TR3$$$_QUAL_MSG             = 00000000
TR3$$$_QUAL_RTFLG           = 00000000
TR3$C_HSZ_DATA              = 00000006
TR3$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
TR3C_MAX_PSW                = 00000040
TR3C_MSG_RT                 = 00000007
TR3C_MSG_RTH                = 00000002
TR3C_MSG_STR                = 00000001
TR3C_MSG_TST                = 00000005
TR3C_MSG_VRF                = 00000003
TR3C_NTY_PH3                = 00000002
TR3C_NTY_PH3N               = 00000003
TR3C_NUM_TST                = 00000003
TR3C_RT_LNG                 = 00000005
TR3C_STR_LNG                = 0000000A
TR3C_STR_RSXL               = 00000009
TR3C_TIVER                  = 00000301
TR3C_TST_MAX                = 0000007F
TR3C_VRF_LNG                = 00000004
TR3C_VRF_MXL                = 00000044
TR3S_REQ_NTY                = 00000002
TR3S_RT_COST                = 0000000A
TR3S_RT_HOPS                = 00000005
TR3V_REQ_NTY                = 00000000
TR3V_REQ_VRF                = 00000002
TR3V_RT_COST                = 00000000
TR3V_RT_HOPS                = 0000000A
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$V_SCLASS_SUBA        =  00000003
TR4C_ART_LNG             =  00000006
TR4C_BCT3MULT            =  00000003
TR4C_EHEL_LNG            =  00000020
TR4C_MAX_PSW             =  00000040
TR4C_MAX_RSLIST          =  000000EC
TR4C_MSG_ART             =  00000009
TR4C_MSG_EHEL            =  0000000D
TR4C_MSG_ENH             =  00000006
TR4C_MSG_RHEL            =  0000000B
TR4C_MSG_RT              =  00000007
TR4C_MSG_STR             =  00000001
TR4C_MSG_VRF             =  00000003
TR4C_NTY_ARO             =  00000001
TR4C_NTY_NROU            =  00000003
TR4C_NTY_ROU             =  00000002
TR4C_RHEL_LNG            =  0000001B
TR4C_RT_LNG              =  00000006
TR4C_STR_LNG             =  0000000C
TR4C_T3MULT              =  00000002
TR4C_TIVER               =  00000002
TR4C_VRF_LNG             =  00000004
TR4C_VRF_MXL             =  00000044
TR4S_REQ_NTY             =  00000002
TR4S_RS_PRIO             =  00000006
TR4S_RT_COST             =  0000000A
TR4S_RT_HOPS             =  00000005
TR4V_REQ_NTY             =  00000000
TR4V_REQ_VRF             =  00000002
TR4V_RS_PRIO             =  00000000
TR4V_RS_TWOWAY           =  00000007
TR4V_RT_COST             =  00000000
TR4V_RT_HOPS             =  0000000A
TR_C_MAX_PSW             =  00000040
TR_C_VRF_LNG             =  00000044
UCB$L_DDT                =  00000088
UCB$L_DEVDEPEND          =  00000044
UCB$W_UNIT               =  00000054
UNK                         000007FC  R     06
UPDATE                      0000149F  R     06
UPDATE_ALL                  00000000  R     07
UPDATE_MATRIX               00001310  R     06
UPDATE_TIMER                00000021  R     06
UPD_NEIGHBORS               0000195F  R     06
WQE$ALLOCATE                ********    X   06
WQE$B_EVL_DT1            =  0000001E
WQE$B_EVT                =  00000010
WQE$CANCEL_TIM              ********    X   06
WQE$C_LENGTH             =  00000024
WQE$C_QUAL_DLL           =  00000001
WQE$C_QUAL_RTG           =  00000002
WQE$C_SUB_ACP            =  00000001
WQE$C_SUB_AST            =  00000003
WQE$DEALLOCATE              ********    X   06
WQE$INSQUE                  ********    X   06
WQE$L_ACTION             =  0000000C
WQE$L_EVL_PKT            =  00000018
WQE$L_PM2                =  00000014
WQE$RESET_TIM               ********    X   07
WQE$W_ADJ_INX            =  00000020
WQE$W_EVL_CODE           =  0000001C
WQE$W_REQIDT             =  00000012
X25_DEACCESS                000022C4  R     06
X25_PVC_SHUTDOWN            000022D4  R     06
X25_SHUTDOWN                000022AF  R     06
X25_STARTUP                 00002423  R     06
XM$V_ERR_FATAL           =  00000010
XM$V_ERR_MAINT           =  00000013
XMT                         000027E0  R     06
XMTFLG                      00000034  R     04
XMT_ART                     000029E3  R     06
XMT_DALLY                   000026C0  R     06
XMT_RT                      0000289D  R     06
XMT_RT4                     00002921  R     06
XMT_STR                     000026EC  R     06
XMT_VRF                     000027FC  R     06
XPT_TO_PTY                  00001B7C  R     06
_$$                      =  00000050
_$ENT                    =  00000000
_$LOG                    =  00020112
_$MAXINX                 =  0000002C
_$TMP                    =  00000000  R     02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
NET_PURE	00000168 (360.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
TABLES_PURE	000004A0 (1184.)	03 (3.)	NOPIC USR CON REL GBL NOSHR NOEXE RD NOWRT NOVEC BYTE
NET_IMPURE	00000042 (66.)	04 (4.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
TABLES_IMPURE	00002B90 (11152.)	05 (5.)	NOPIC USR CON REL GBL NOSHR NOEXE RD WRT NOVEC LONG
NET_CODE	00002D45 (11589.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE
NET_LOCK_CODE	000000BD (189.)	07 (7.)	NOPIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.11	00:00:00.68
Command processing	124	00:00:01.04	00:00:04.35
Pass 1	2249	00:01:17.81	00:01:58.31
Symbol table sort	2	00:00:05.87	00:00:06.84
Pass 2	1521	00:00:22.39	00:00:35.82
Symbol table output	1	00:00:00.66	00:00:00.78
Psect synopsis output	4	00:00:00.04	00:00:00.12
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	3934	00:01:47.95	00:02:46.98

The working set limit was 1650 pages.
418475 bytes (818 pages) of virtual memory were used to buffer the intermediate code.
There were 210 pages of symbol table space allocated to hold 3408 non-local and 643 local symbols.
7889 source lines were read in Pass 1, producing 70 object records in Pass 2.
76 pages of virtual memory were used to define 64 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]NMALIBRY.MLB;1	1
_255DUA28:[SHRLIB]EVCDEF.MLB;1	1
_255DUA28:[NETACP.OBJ]NETDRV.MLB;1	1
_255DUA28:[NETACP.OBJ]NET.MLB;1	20
_255DUA28:[SYS.OBJ]LIB.MLB;1	8
_255DUA28:[SYSLIB]STARLET.MLB;2	16
TOTALS (all libraries)	47

3429 GETS were required to define 47 macros.

There were no errors, warnings or information messages.

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

NETDLLTRN
LIS

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

NETDRVQRL
LIS

NETDRVSES
LIS

NETDRVNSP
LIS