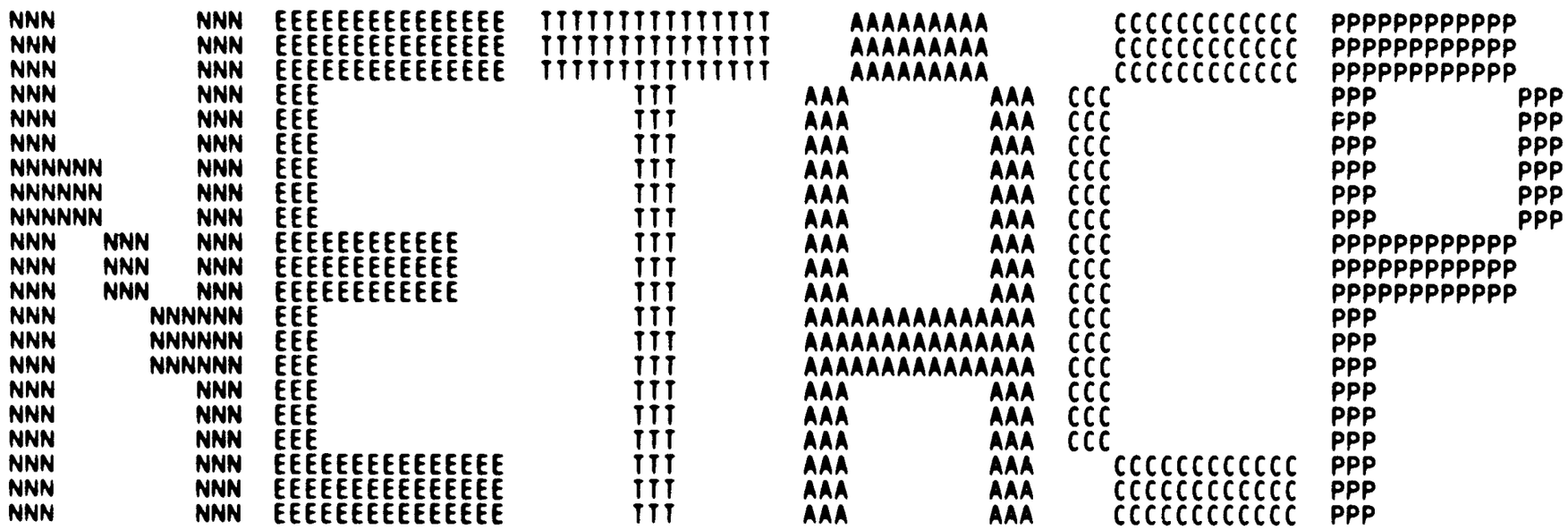
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN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

```
NN      NN  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    RRRRRRRR    VV      VV    QQQQQQ    RRRRRRRR    LL
NN      NN  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    RRRRRRRR    VV      VV    QQQQQQ    RRRRRRRR    LL
NN      NN  EE              TT      DD      DD  RR      RR  VV      VV  QQ      QQ  RR      RR  LL
NN      NN  EE              TT      DD      DD  RR      RR  VV      VV  QQ      QQ  RR      RR  LL
NNNN    NN  EE              TT      DD      DD  RR      RR  VV      VV  QQ      QQ  RR      RR  LL
NNNN    NN  EE              TT      DD      DD  RR      RR  VV      VV  QQ      QQ  RR      RR  LL
NN  NN  NN  EEEEEEEE        TT      DD      DD  RRRRRRRR    VV      VV  QQ      QQ  RRRRRRRR    LL
NN  NN  NN  EEEEEEEE        TT      DD      DD  RRRRRRRR    VV      VV  QQ      QQ  RRRRRRRR    LL
NN    NNNN  EE              TT      DD      DD  RR  RR      VV      VV  QQ  QQ  QQ  RR  RR      LL
NN    NNNN  EE              TT      DD      DD  RR  RR      VV      VV  QQ  QQ  QQ  RR  RR      LL
NN      NN  EE              TT      DD      DD  RR    RR      VV  VV    QQ    QQ    RR    RR    LL                  ....
NN      NN  EE              TT      DD      DD  RR    RR      VV  VV    QQ    QQ    RR    RR    LL                  ....
NN      NN  EEEEEEEEEE      TT      DDDDDDDD    RR      RR      VV      QQQQ  QQ  RR      RR  LLLLLLLLLL          ....
NN      NN  EEEEEEEEEE      TT      DDDDDDDD    RR      RR      VV      QQQQ  QQ  RR      RR  LLLLLLLLLL          ....

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
```

```
0000     1 ;& - Someday, the LPE will be supported as part of the LPD
0000     2
0000     3         .TITLE  NETDRVQRL - DECnet 'Quick Routing Layer' module for NETDRIVER
0000     4         .IDENT  'V04-000'
0000     5 ;
0000     6 ;*****************************************************************************
0000     7 ;*                                                                           *
0000     8 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     9 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    10 ;*  ALL RIGHTS RESERVED.                                                     *
0000    11 ;*                                                                           *
0000    12 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    13 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    14 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    15 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    16 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    17 ;*  TRANSFERRED.                                                             *
0000    18 ;*                                                                           *
0000    19 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    20 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    21 ;*  CORPORATION.                                                             *
0000    22 ;*                                                                           *
0000    23 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    24 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    25 ;*                                                                           *
0000    26 ;*                                                                           *
0000    27 ;*****************************************************************************
0000    28
0000    29 ;++
0000    30 ; FACILITY:     DECnet, Executive
0000    31 ;
0000    32 ; ABSTRACT:
0000    33 ;               This module implements a quick interface for high speed
0000    34 ;               communications in end-node environments where the partner
0000    35 ;               node is 1-hop away, e.g., Ethernet environments.  The
0000    36 ;               motivation for it is the inordinate amount of time spent in
0000    37 ;               the more general purpose NETDRVXPT module.
0000    38 ;
0000    39 ; ENVIRONMENT:  Standard driver environment
0000    40 ;
0000    41 ;--
0000    42
```

```
0000     44          .SBTTL  MODIFICATION HISTORY
0000     45 ;
0000     46 ; AUTHOR:       Alan D. Eldridge,  CREATION DATE: 30-Oct-1983
0000     47 ;
0000     48 ; MODIFIED BY:
0000     49 ;
0000     50 ;       V04-001 RNG0001                 Rod Gamache     19-Mar-1984
0000     51 ;               Close off call to QRL$SETUP_CHAN.
0000     52 ;
0000     53 ;
0000     54
```

```
0000     56          .SBTTL  DECLARATIONS
0000     57 ;
0000     58 ; INCLUDE FILES:
0000     59 ;
0000     60          $CXBDEF
0000     61          $DYNDEF
0000     62          $FKBDEF
0000     63          $IPLDEF
0000     64          $IRPDEF
0000     65          $IODEF
0000     66          $SSDEF
0000     67          $TQEDEF
0000     68          $UCBDEF
0000     69          $VADEF
0000     70
0000     71          $ADJDEF
0000     72          $LPDDEF
0000     73          $RCBDEF
0000     74
0000     75          $NETSYMDEF
0000     76          $NETUPDDEF
0000     77          $NSPMSGDEF
0000     78
0000     79          $CXBEXTDEF              ; NETDRIVER CXB extensions
0000     80          $XWBDEF                 ; XWB and LSB definitions
0000     81
0000     82 .iif ndf,IRP$Q_STATION, IRP$Q_STATION = 8+IRP$L_MEDIA
0000     83
0000     84
```

```
                     0000    86
                     0000    87 ;
                     0000    88 ;  LOCAL DEFINITIONS
                     0000    89 ;
0000000F             0000    90 HSZ_DELTA = TR4$C_HSZ_DATA-TR3$C_HSZ_DATA       ; Difference in header sizes
                     0000    91
00000008             0000    92 MAX_C_LPE = 8                                   ; Max LPE's
                     0000    93
                     0000    94
                     0000    95
00000000             0000    96         LPE$Q_IRP_WAIT  =  0                    ; Listhead of waiting processes
00000008             0000    97         LPE$Q_IRP_FREE  =  8                    ; Listhead of free IRP's
00000010             0000    98         LPE$B_IRP_CNT   = 16                    ; Count of current IRP's
00000014             0000    99         LPE$C_LENGTH    = 20                    ; Round up the length
                     0000   100
                     0000   101 ;
                     0000   102 ;  MACROS
                     0000   103 ;
00000001             0000   104 CA$_MEASURE = 1 ;&
                     0000   105
                     0000   106 .MACRO  INCPMS  PMS_CELL                        ; Increment PMS cell
                     0000   107                                                 ;
                     0000   108         .IF DF  CA$_MEASURE                     ;
                     0000   109         .IF NE  CA$_MEASURE                     ; Conditional assembly
                     0000   110                 INCL G^PMS$GL_'PMS_CELL'        ; Bump the counter
                     0000   111         .ENDC                                   ;
                     0000   112         .ENDC                                   ;
                     0000   113 .ENDM   INCPMS                                  ;
                     0000   114
                     0000   115
                     0000   116
                 00000000   117         .PSECT  $$$115_DRIVER,LONG,EXE,RD,WRT   ; Goto code PSECT
                     0000   118
                     0000   119
                     0000   120
```

```
                     0000   122          .SBTTL  QRL$SOLICIT     - Process ECL request to xmit into the network
                     0000   123 ;+
                     0000   124 ;
                     0000   125 ;  An ECL (e.g. NSP) is requesting to xmit into the network.  The request is
                     0000   126 ;  honored as soon as an IRP for the designated IRP is available.
                     0000   127 ;
                     0000   128 ;  INPUTS:       R5      Fork block address
                     0000   129 ;                          The FPC,FR3,FR4 fields are all scratch and must not
                     0000   130 ;                          be modified by the caller until it is reactivated by
                     0000   131 ;                          either  QRL$DENY  or  QRL$GRANTED.
                     0000   132 ;                  R4      ADJ index
                     0000   133 ;                  R3      LPD i.d.
                     0000   134 ;                  R2      RCB address
                     0000   135 ;                  R1,R0   Scratch
                     0000   136 ;
                     0000   137 ;                  (SP)    Return address of caller
                     0000   138 ;                 4(SP)    Return address of caller's caller
                     0000   139 ;
                     0000   140 ;
                     0000   141 ;  OUTPUTS:      See routines QRL$GRANT or QRL$DENY
                     0000   142 ;
                     0000   143 ;-
                     0000   144 QRL$SOLICIT::                                    ; Process ECL request to xmit
                     0000   145          ;
                     0000   146          ;
                     0000   147          ;    Setup the fork block and pop the stack to simplify the code
                     0000   148          ;    in case the requestor needs to be suspended.
                     0000   149          ;
                     0000   150          ;
                0C A5 8ED0  0000   151          POPL    FKB$L_FPC(R5)               ; Save return addr, pop stack
              07C0 8F   BB  0004   152          PUSHR   #^M<R6,R7,R8,R9,R10>        ; Save regs
                     0008   153                                                      ;
              53   53   9A  0008   154          MOVZBL  R3,R3                       ; Use only the index
            10 A5   53   7D  000B   155          MOVQ    R3,FKB$L_FR3(R5)            ; Save selection data
         58   28 B243   D0  000F   156          MOVL    @RCB$L_PTR_LPD(R2)[R3],R8   ; Get LPD address
                 05   10  0014   157          BSBB    10$                         ; Process request, okay to wait
                     0016   158                                                      ;
              07C0 8F   BA  0016   159          POPR    #^M<R6,R7,R8,R9,R10>        ; Restore regs
                 05        001A   160          RSB                                 ; Done
                     001B   161
                     001B   162
                     001B   163 10$:     ASSUME  LPD$V_ACTIVE  EQ  0
                     001B   164
           16 22 A8   E9  001B   165          BLBC    LPD$W_STS(R8),QRL$DENY      ; Br if no path to node
                     001F   166 QUICK_SOL:                                         ; Quick solicit entry
         56   53   14   C5  001F   167          MULL3   #LPE$C_LENGTH,R3,R6         ; Get LPE offset
   56   00000000'GF   C0  0023   168          ADDL    G^NET$GCB,R6                ; Make it a pointer
                     002A   169          ;
                     002A   170          ;
                     002A   171          ;    Get LPD specific IRP -- also serves as the 'input packet limiter'
                     002A   172          ;
                     002A   173          ;
            53   08 B6   0F  002A   174          REMQUE  @LPE$Q_IRP_FREE(R6),R3      ; Get a free IRP
                 11   1C  002E   175          BVC     QRL$GRANT                   ; If VC then got one
            04 B6   65   0E  0030   176          INSQUE  (R5),@LPE$Q_IRP_WAIT+4(R6)  ; Wait for IRP
                 05        0034   177          RSB
                     0035   178
```

```
                     0035   180          .SBTTL  QRL$DENY        - Deny solicitor permission to transmit
                     0035   181          .SBTTL  QRL$GRANT       - Grant solicitor permission to transmit
                     0035   182 ;+
                     0035   183 ;
                     0035   184 ;  The R5 fork process cannot be suspended beyond this point.
                     0035   185 ;
                     0035   186 ;
                     0035   187 ;  INPUTS:      R9      Scratch
                     0035   188 ;               R8      LPD address
                     0035   189 ;               R7,R6   Scratch
                     0035   190 ;               R5      Fork block address
                     0035   191 ;               R4      ADJ index
                     0035   192 ;               R3      If QRL$GRANT  - IRP address
                     0035   193 ;                       If QRL$DENY   - Scratch
                     0035   194 ;               R2      RCB address
                     0035   195 ;               R1,R0   Scratch
                     0035   196 ;
                     0035   197 ;
                     0035   198 ;  OUTPUTS:     R7-R0   Garbage
                     0035   199 ;
                     0035   200 ;               All other registers are preserved.
                     0035   201 ;
                     0035   202 ;-
                     0035   203 QRL$DENY:                                    ; Deny permisson to xmit
          50   94    0035   204          CLRB    R0                         ; Indicate request denied
       8E 52   7D    0037   205          MOVQ    R2,(SP)+                   ; Save regs
     0C B5   16      003A   206          JSB     @FKB$L_FPC(R5)             ; Tell requestor the bad news
    52 8E    7D      003D   207          MOVQ    (SP)+,R2                   ; Restore regs
             05      0040   208          RSB                                ; Done
                     0041   209
                     0041   210 QRL$GRANT:                                  ; Grant permission to xmit
                     0041   211          ;
                     0041   212          ;    Call requestor back with:
                     0041   213          ;
                     0041   214          ;       R9      ADJ address
                     0041   215          ;       R8      LPD address
                     0041   216          ;       R7,R6   Scratch
                     0041   217          ;       R5      Fork block address
                     0041   218          ;       R4      Scratch
                     0041   219          ;       R3      IRP address only if R0 has low bit set, else scratch
                     0041   220          ;       R2      RCB address
                     0041   221          ;       R1      Scratch
                     0041   222          ;       R0      Low bit set if permission granted
                     0041   223          ;               Low bit clear if permission denied
                     0041   224          ;
                     0041   225          ;
        59   2C B244  D0     0041   226          MOVL    @RCB$L_PTR_ADJ(R2)[R4],R9          ; Get ADJ
                     0046   227 ;&        MOVW    R10,IRP$Q_STATION+4(R3)            ; Store dest node addr
                     0046   228 ;&        BBC     #ADJ$V_RUN,ADJ$B_STS(R9),10$       ; If BC, "main" ADJ
      44 A3   04 A9  3C     0046   229          MOVZWL  ADJ$W_PNA(R9),IRP$Q_STATION+4(R3)  ; Set target address
40 A3  000400AA 8F    D0     004B   230 10$:     MOVL    #TR4$C_HIORD,IRP$Q_STATION(R3)     ; ...
            50   01    90     0053   231          MOVB    #1,R0                      ; Say "okay to xmit"
           0C B5   16      0056   232          JSB     @FKB$L_FPC(R5)             ; Reactivate solicitor
                     0059   233          ;
                     0059   234          ;
                     0059   235          ;    On return, the CXB and registers are setup as follows:
                     0059   236          ;
```

```
                                ;
                                ;   +---------------+
                                ;   |   standard    |   11 bytes long.  CXB$L_FLINK and CXB$L_BLINK may
                                ;   |     VMS       |   be used by the Transport layer.  CXB$W_SIZE
                                ;   | buffer header |   must be correct.  CXB$B_TYPE must be DYN$C_CXB.
                                ;   +---------------+
                                ;   |               |   Starts with CXB$B_CODE (byte 11) and continues
                                ;   |     ECL       |   to CXB$C_LENGTH.  This area is read-only to
                                ;   |   pure area   |   Transport and below.  It cannot even be
                                ;   |               |   saved/restored.
                                ;   +---------------+
                                ;   |   Datalink    |   Starts at CXB$C_LENGTH and is at least
                                ;   |     Layer     |   CXB$C_DLL bytes long.  Used by the datalink for
                                ;   |  impure area  |   protocol header or state information.
                                ;   +---------------+
                                ;   |               |
                                ;   .   body of     .   Must be quadword aligned and starting no sooner
                                ;   .   message     .   than CXB$C_LENGTH + CXB$C_DLL (= CXB$C_HEADER)
                                ;   .               .   The first 6 bytes contain: RTFLG,DSTNOD,SRCNOD
                                ;   |               |   FORWARD, in that order.
                                ;   +---------------+
                                ;   |   Datalink    |   Used by the datalink layer for protocol (e.g.,
                                ;   |     Layer     |   checksum) or state information.  Must be at
                                ;   |  impure area  |   least CXB$C_TRAILER in length.
                                ;   +---------------+
                                ;
                                ;       R9      ADJ address or zero
                                ;       R8      LPD address
                                ;       R7      Size of message
                                ;       R6      CXB address
                                ;       R5      Garbage
                                ;       R4      0 if "quick solicit" not requested
                                ;               Else, pointer to request block (XWB fork block) with
                                ;               FRK$L_FPC  pointing to the "quick solicit" routine
                                ;       R3      IRP address -- unmodified from call
                                ;       R2      Address of End-action routine to call on I/O competion
                                ;       R1      Ptr to 1st byte in standard Phase III route-header
                                ;       R0      Low bit set   - if message is to be xmitted
                                ;               Low bit clear - if no message to xmit.  In this case
                                ;                               R7-R4,R2,R1 contain garbage.
                                ;
                                ;
78 A3    52    D0   0059   281 MOVL    R2,IRP$L_SAVD_RTN(R3)   ; Save ptr to End-action routine
52    14 A3    D0   005D   282 MOVL    IRP$L_ASTPRM(R3),R2     ; Recover RCB address
      11 50    E9   0061   283 BLBC    R0,60$                  ; If LBC then xmit aborted
                    0064   284                                  ;
   55    54    D0   0064   285 MOVL    R4,R5                   ; "Quick solicit" requested ?
         0F    13   0067   286 BEQL    FINISH_XMT              ; If EQL, no
                    0069   287                                  ;
         24    BB   0069   288 PUSHR   #^M<R2,R5>              ; Remember RCB and block's address
         0B    10   006B   289 BSBB    FINISH_XMT              ; Finish building HDR & IRP, xmit it.
         24    BA   006D   290 POPR    #^M<R2,R5>              ; Setup R2,R5  (R8 points to LPD)
                    006F   291                                  ;
53    10 A5    7D   006F   292 MOVQ    FKB$L_FR3(R5),R3        ; Recover original request data
         AA    11   0073   293 BRB     QUICK_SOL               ; Perform "quick solicit"
```

```
                00B1   31   0075   294 60$:    BRW     QRL$XMT_ABORTED          ; Recycle unused the IRP
                            0078   295
                            0078   296 FINISH_XMT:
                            0078   297
                            0078   298         ;
                            0078   299         ;
                            0078   300         ;    Journal the message to be transmitted
                            0078   301         ;
                            0078   302         ;
                            0078   303 ;       .IF     DF,JNX$$$
                            0078   304 ;
                            0078   305 ;       PUSHL   R0                       ; Save registers
                            0078   306 ;       MOVL    IRP$L_ASTPRM(R3),R2      ; Get RCB address
                            0078   307 ;       CLRL    R0                       ; Set journal type = Start xmt
                            0078   308 ;       BSBW    TR_FILL_JNX              ; Store journal record
                            0078   309 ;       POPL    R0                       ; Restore registers
                            0078   310 ;
                            0078   311 ;       .ENDC
                            0078   312
          0D 22 A8    07    E1    0078   313         BBC     #LPD$V_X25,LPD$W_STS(R8),100$   ; Skip if not X.25 datalink
                            007D   314         ;
                            007D   315         ;
                            007D   316         ;    X.25 circuits a CRC16 checksum on the data portion of the message.
                            007D   317         ;
                            007D   318         ;
                      0B    BB    007D   319         PUSHR   #^M<R0,R1,R3>            ; Save regs
                            007F   320 ;&      CRC     CRC16,#0,R7,(R1)         ; Calculate CRC16 on data
                52    50    D0    007F   321         MOVL    R0,R2                    ; Save CRC
                      0B    BA    0082   322         POPR    #^M<R0,R1,R3>            ; Restore regs
                            0084   323         ;
                71    52    B0    0084   324         MOVW    R2,-(R1)                 ; Save CRC in datagram
                57    02    A0    0087   325         ADDW    #2,R7                    ; Account
                66    51    D0    008A   326 100$:    MOVL    R1,(R6)                  ; Save address of start of data
                            008D   327         ;
              20 A3   20    B0    008D   328         MOVW    S^#IO$_WRITELBLK,IRP$W_FUNC(R3) ; Setup function
              24 A3   56    D0    0091   329         MOVL    R6,IRP$L_IOSB(R3)        ; Buffer address into IOSB
                            0095   330         ;
                            0095   331         ASSUME  IRP$W_BOFF  EQ  4+IRP$L_SVAPTE
                            0095   332         ;
              50    2C A3   9E    0095   333         MOVAB   IRP$L_SVAPTE(R3),R0      ; Setup for auto-increment
          07 22 A8    05    E1    0099   334         BBC     #LPD$V_XBF,LPD$W_STS(R8),120$   ; If BC,  I/O is direct
                            009E   335         ;
                            009E   336         ;
                            009E   337         ;    Xmitter I/O is buffered
                            009E   338         ;
                            009E   339         ;
                80    56    D0    009E   340         MOVL    R6,(R0)+                 ; Setup buffer ptr in SVAPTE
                      80    B4    00A1   341         CLRW    (R0)+                    ; Clear BOFF
                      19    11    00A3   342         BRB     140$                     ; Continue
                            00A5   343 120$:    ;
                            00A5   344         ;
                            00A5   345         ;    Xmitter I/O is direct
                            00A5   346         ;
                            00A5   347         ;
                54    66    D0    00A5   348         MOVL    (R6),R4                  ; Get msg address
           56   00000000'GF    D0    00A8   349         MOVL    G^MMG$GL_SPTBASE,R6      ; Get system page table base
         51   54    15    09    EF    00AF   350         EXTZV   #VA$V_VPN,#VA$S_VPN,R4,R1 ; Get Virtual page frame number
```

```
                80   6641   DE  00B4   351          MOVAL   (R6)[R1],(R0)+                    ; Enter SVAPTE
          80    54   FE00 8F  AB  00B8   352          BICW3   #^C<VA$M_BYTE>,R4,(R0)+          ; Enter page offset into BOFF
                                   00BE   353 140$:   ;
                                   00BE   354         ;
                                   00BE   355         ;    Complete the IRP and queue it to the device
                                   00BE   356         ;
                                   00BE   357         ;
                                   00BE   358         ASSUME  IRP$W_BCNT   EQ  2+IRP$W_BOFF
                                   00BE   359         ASSUME  IRP$L_BCNT   EQ  0+IRP$W_BCNT
                                   00BE   360
                    60   57   3C  00BE   361         MOVZWL  R7,(R0)                          ; Enter BCNT
                         56   D4  00C1   362         CLRL    R6                               ; Prevent buffer deallocation
                    55   1C A3   D0  00C3   363         MOVL    IRP$L_UCB(R3),R5                 ; Get comm driver UCB
                         06   13  00C7   364         BEQL    150$                             ; If EQL then this is Local LPD
                00000000'GF   17  00C9   365         JMP     G^EXE$ALTQUEPKT                  ; Send to "real" datalink
                                   00CF   366
                                   00CF   367 ;150$:  BRW     TR$LOC_DLL_XMT                   ; Send to "local" datalink
                                   00CF   368
                                   00CF   369 150$:   BUG_CHECK  NETNOSTATE,FATAL             ; TR$LOC_DLL_XMT not global
                                   00D3   370
                                   00D3   371
```

```
                        00D3    373
                        00D3    374 QRL$RTRN_X_IO:                              ; Xmt I/O completion
                        00D3    375         DSBINT  #NET$C_IPL                  ; Raise IPL
        07C0 8F     BB  00D9    376         PUSHR   #^M<R6,R7,R8,R9,R10>        ; Save regs
                        00DD    377                                             ;
    52    14 A5     D0  00DD    378         MOVL    IRP$L_ASTPRM(R5),R2         ; Get RCB pointer
    50    10 A5     9A  00E1    379         MOVZBL  IRP$L_AST(R5),R0            ; Get LPD index
58    28 B240     D0  00E5    380         MOVL    @RCB$L_PTR_LPD(R2)[R0],R8   ; Get LPD address
                        00EA    381
                        00EA    382 ;&      .IF     DF,JNX$$$
                        00EA    383 ;
                        00EA    384 ;       MOVL    #8,R7                       ; Set length of IOSB
                        00EA    385 ;       MOVAB   IRP$L_IOST1(R5),R1          ; Journal the IOSB quadword
                        00EA    386 ;       MOVB    #1,R0                       ; Journal type = xmt complete
                        00EA    387 ;       BSBW    TR_FILL_JNX                 ; Store journal record
                        00EA    388 ;
                        00EA    389 ;&      .ENDC
                        00EA    390
    50    24 A5     D0  00EA    391         MOVL    IRP$L_IOSB(R5),R0           ; Get buffer
          24 A5     D4  00EE    392         CLRL    IRP$L_IOSB(R5)              ; Detach it from the IRP
                        00F1    393         ;
                        00F1    394         ;
                        00F1    395         ;   Deliver end-action status to the ECL issuing the transmit.  It is
                        00F1    396         ;   the responsibility of the ECL routine to consume the R0 buffer --
                        00F1    397         ;   deallocate it, requeue it, etc.  Attaching R0 to IRP$L_IOSB will
                        00F1    398         ;   cause it to be deallocated on return.
                        00F1    399         ;
                        00F1    400         ;
                        00F1    401         ;   Call with: R5      IRP address
                        00F1    402         ;               R4,R3   Scratch
                        00F1    403         ;               R2      RCB address
                        00F1    404         ;               R1      Scratch
                        00F1    405         ;               R0      CXB address
                        00F1    406         ;
                        00F1    407         ;
                        00F1    408         ;   On return from ECL:
                        00F1    409         ;
                        00F1    410         ;               R4,R3,R1,R0     may be garbage.
                        00F1    411         ;
                        00F1    412         ;               All other registers must be unchanged.
                        00F1    413         ;
                        00F1    414         ;
        78 B5     16    00F1    415         JSB     @IRP$L_SAVD_RTN(R5)         ; Deliver status to ECL layer
                        00F4    416         INCPMS  RCVBUFFL                ;& temp
    0F 38 A5      E9    00FA    417         BLBC    IRP$L_IOST1(R5),30$         ; If LBC, I/O failed
                        00FE    418         BUMP    L,LPD$L_CNT_DPS(R8)         ; Bump 'departing pkts sent'
                        0107    419         INCPMS  DEPLOCPR                    ; ... and the PMS database too
50    24 A5     D0  010D    420 30$:    MOVL    IRP$L_IOSB(R5),R0           ; Get the CXB
          09     13  0111    421         BEQL    40$                         ; If EQL then none
          24 A5     D4  0113    422         CLRL    IRP$L_IOSB(R5)              ; Nullify CXB pointer
00000000'GF     16  0116    423         JSB     G^COM$DRVDEALMEM            ; Deallocate the CXB
    53    55     D0  011C    424 40$:    MOVL    R5,R3                       ; Copy IRP address
          0C     10  011F    425         BSBB    QRL$RTRN_X_IRP              ; Return IRP to the Xmit pool
                        0121    426                                             ;
    07C0 8F     BA  0121    427         POPR    #^M<R6,R7,R8,R9,R10>        ; Restore reg
                        0125    428         ENBINT                              ; Restore IPL
                  05    0128    429         RSB                                 ; Return to Exec
```

0129 430

```
                     0129   432 QRL$XMT_ABORTED:                                ; User abort Xmit request
        38 A3   01   90 0129   433          MOVB    #1,IRP$L_IOST1(R3)          ; Recycle IRP normally
                     012D   434                                                  ;
                     012D   435 QRL$RTRN_X_IRP:                                  ; Recycle Xmt the IRP
        50   10 A3   9A 012D   436          MOVZBL  IRP$L_AST(R3),R0            ; Get LPD index
     56   50   14   C5 0131   437          MULL3   #LPE$C_LENGTH,R0,R6         ; Get LPE offset
  56  00000000'GF   C0 0135   438          ADDL    G^NET$GCB,R6                ; Make it a pointer
        52   14 A3   D0 013C   439          MOVL    IRP$L_ASTPRM(R3),R2         ; Get RCB pointer
                     0140   440          ;
                     0140   441          ;
                     0140   442          ;    Restart someone waiting for this IRP, or queue the IRP.
                     0140   443          ;
                     0140   444          ;
        55   00 B6   0F 0140   445 40$:    REMQUE  @LPE$Q_IRP_WAIT(R6),R5      ; Get waiting process
                10   1D 0144   446          BVS     70$                         ; If VS, none
          07 38 A3   E9 0146   447          BLBC    IRP$L_IOST1(R3),50$         ; If LBC, I/O was unsuccessful
        54   14 A5   D0 014A   448          MOVL    FKB$L_FR4(R5),R4            ; Get ADJ index
              FEF0   31 014E   449          BRW     QRL$GRANT                   ; Reactivate the process
                     0151   450 50$:    ;
                     0151   451          ;
                     0151   452          ;    An I/O failure indicates a severe hardware problem.  Reactivate
                     0151   453          ;    all processes waiting for an IRP for this device denying them
                     0151   454          ;    permission to transmit.  The ECL recovery logic will try again
                     0151   455          ;    after a short interval.  The I/O failure will be detected in
                     0151   456          ;    NETDRVXPT since it always has a receive posted to each circuit
                     0151   457          ;    and that receive should also fail.
                     0151   458          ;
                     0151   459          ;
              FEE1   30 0151   460          BSBW    QRL$DENY                    ; Reactivate the process
                EA   11 0154   461          BRB     40$                         ; Loop
        0C B6   63   0E 0156   462 70$:    INSQUE  (R3),@LPE$Q_IRP_FREE+4(R6)  ; Save IRP
                     05 015A   463          RSB
                     015B   464
```

```
                     015B   466          .SBTTL  QRL$SETUP_CHAN  - Setup channel to specified node
                     015B   467 ;+
                     015B   468 ;
                     015B   469 ;  INPUTS:      R3      Non-zero LPD index
                     015B   470 ;               R2      RCB pointer
                     015B   471 ;               R1      Pointer to standard Phase III route-header
                     015B   472 ;
                     015B   473 ;  OUTPUS:      R4      ADJ index
                     015B   474 ;               R3      Size of new route-header
                     015B   475 ;               R1      Pointer to new route-header
                     015B   476 ;               R0      LBS if successful, LBC otherwise
                     015B   477 ;
                     015B   478 ;-
                     015B   479 QRL$SETUP_CHAN::                                     ; Setup channel to node
 03F6 8F     BB      015B   480          PUSHR   #^M<R1,R2,R4,R5,R6,R7,R8,R9>       ; Save regs
      50     D4      015F   481          CLRL    R0                                 ; Assume error
 07   53     91      0161   482          CMPB    R3,#MAX_C_LPE-1                    ; Can we handle it?
                     0164   483 ;;&&      BGEQU   5$                                 ; If GEQU, out of range
      0A     11      0164   484          BRB     5$                                 ; Always return failure
                     0166   485          ;
                     0166   486          ;
                     0166   487          ;    Find ADJ and LPD
                     0166   488          ;
                     0166   489          ;
 54   01 A1   3C     0166   490          MOVZWL  1(R1),R4                           ; Get remote node address
        FE93'  30     016A   491          BSBW    TR$GET_ADJ                         ; Get ADJ and LPD
      03 50   E8      016D   492          BLBS    R0,10$                             ; If LBS okay
       007C   31      0170   493 5$:      BRW     100$                               ; Exit
                     0173   494                                                     ;& is the following needed?
            50   D4   0173   495 10$:     CLRL    R0                                 ; Assume LPD or ADJ not there
      76 58    1F     E1     0175   496          BBC     #31,R8,100$                ; If BC, no LPD
      72 59    1F     E1     0179   497          BBC     #31,R9,100$                ; If BC, no ADJ
      50   2C A2      D0     017D   498          MOVL    RCB$L_PTR_ADJ(R2),R0       ; Address first ADJ pointer
         80    59     D1     0181   499 20$:     CMPL    R9,(R0)+                   ; This it ?
               FB     12     0184   500          BNEQ    20$                        ; If EQL, no
      50   2C A2      C2     0186   501          SUBL    RCB$L_PTR_ADJ(R2),R0       ; Get difference in longwords
         50    04     C6     018A   502          DIVL    #4,R0                      ; Convert to bytes
 08 AE   50    01     C3     018D   503          SUBL3   #1,R0,8(SP)                ; Save index R4 cell in stack
                     0192   504          ;
                     0192   505          ;
                -    0192   506          ;    Make sure LPE has IRP's.   If LPE's don't exist, allocate them.
                     0192   507          ;
                     0192   508          ;
56   00000000'GF     D0     0192   509          MOVL    G^NET$WCB,R6               ; Get LPE vector pointer
               2C    12     0199   510          BNEQ    50$                        ; If NEQ, it exits
      51   AC 8F     9A     019B   511          MOVZBL  #12+<LFE$C_LENGTH*MAX_C_LPE>,R1 ; Setup length of LPE vector
          FE5E'  30     019F   512          BSBW    NET$ALONPGD_Z              ; Allocate /zero the block
        4A 50   E9      01A2   513          BLBC    R0,100$                    ; If LBC, nice try
   56   52   0C     C1     01A5   514          ADDL3   #12,R2,R6                  ; Get address of vector area
00000000'GF   56     D0     01A9   515          MOVL    R6,G^NET$WCB               ; Setup LPE vector pointer
                     01B0   516
                     01B0   517
                     01B0   518          ASSUME  LPE$Q_IRP_FREE  EQ  8+LPE$Q_IRP_WAIT
                     01B0   519          ASSUME  LPE$C_LENGTH    EQ  20
                     01B0   520
      50   56     D0     01B0   521          MOVL    R6,R0                      ; Setup temp pointer
      52   08     9A     01B3   522          MOVZBL  #MAX_C_LPE,R2              ; Setup loop counter
```

```
                60    50    D0   01B6    523 40$:    MOVL    R0,(R0)                         ; Setup first listhead
                80    80    DE   01B9    524         MOVAL   (R0)+,(R0)+                     ;
                60    50    D0   01BC    525         MOVL    R0,(R0)                         ; Setup second listhead
                80    80    D0   01BF    526         MOVL    (R0)+,(R0)+                     ;
                      80    D5   01C2    527         TSTL    (R0)+                           ; Go to next LPE
                  EF  52    F5   01C4    528         SOBGTR  R2,40$                          ; Loop
                            01C7    529                                                     ;
                53    53    9A   01C7    530 50$:    MOVZBL  R3,R3                           ; Use only the index portion
             56 53    14    C5   01CA    531         MULL3   #LPE$C_LENGTH,R3,R6             ; Get LPE offset
          56  00000000'GF    C0   01CE    532         ADDL    G^NET$QCB,R6                    ; Make it a pointer
                51    6E    7D   01D5    533         MOVQ    (SP),R1                         ; Recover header ptr and RCB
                   0099     30   01D8    534         BSBW    QRL$SETUP_X_IRP                 ; Setup IRP's
                  11 50     E9   01DB    535         BLBC    R0,100$                         ; If LBC, something's wrong
                            01DE    536                 ;
                            01DE    537                 ;
                            01DE    538                 ;    Setup route-header
                            01DE    539                 ;
                            01DE    540                 ;
                51    6E    7D   01DE    541         MOVQ    (SP),R1                         ; Recover header ptr and RCB
                57    06    D0   01E1    542         MOVL    #6,R7                           ; Standard header size
                      0E    10   01E4    543         BSBB    QRL$SETUP_RTHDR                 ; Setup the header
                6E    51    D0   01E6    544         MOVL    R1,(SP)                         ; Setup new header pointer
                53    57    D0   01E9    545         MOVL    R7,R3                           ; Setup new header size
                50    01    D0   01EC    546         MOVL    #1,R0                           ; Say "success"
                            01EF    547                                                     ;
                03F6  8F    BA   01EF    548 100$:   POPR    #^M<R1,R2,R4,R5,R6,R7,R8,R9>    ; Restore regs
                            05   01F3    549         RSB                                     ; Return status in R0
                            01F4    550
```

```
                        .SBTTL  QRL$SETUP_RTHDR - Build route-header
                ;+
                ;
                ;  This routine converts the Phase III header passed by R1 to the proper format
                ;  according to the nature of the adjancency and the padding requirements of
                ;  the circuit.
                ;
                ;  INPUTS:      R9      ADJ address
                ;               R8      LPD address
                ;               R7      Total number of bytes in message/header
                ;               R2      RCB address
                ;               R1      Pointer to start of Phase III route-header
                ;
                ;  OUTPUTS:     R7      New message/header size
                ;               R4,R3   Garbage
                ;               R1      Pointer to start of new route-header
                ;               R0      Garbage
                ;
                ;               All other registers are preserved.
                ;
                ;-
                QRL$SETUP_RTHDR:                                ; Build/convert route-header
                        ;
                        ;
                        ;   Build header based on output adjacency node type.
                        ;
                        ;   We will make a special check here, to see if we are an Endnode.
                        ;   This is because on a BC circuit the "main" ADJ has a PTYPE of
                        ;   "unknown" which prevents the building of a route header.
                        ;
                        ;
                        $DISPATCH RCB$B_ETY(R2),TYPE=B,-
                        <-
                                <ADJ$C_PTY_PH4N, 10$>,-         ; Phase IV endnode
                        >
                        $DISPATCH ADJ$B_PTYPE(R9),TYPE=B,-
                        <-
                                <ADJ$C_PTY_AREA 10$>,-          ; Phase IV level 2 router
                                <ADJ$C_PTY_PH4  10$>,-          ; Phase IV router
                                <ADJ$C_PTY_PH4N 10$>,-          ; Phase IV endnode
                                <ADJ$C_PTY_PH3  30$>,-          ; Phase III router
                                <ADJ$C_PTY_PH3N 30$>,-          ; Phase III endnode
                        >
                        ;
                        ;
                        ;   All others including Phase II
                        ;
                        ;
57  06  C2              SUBL    #TR3$C_HSZ_DATA,R7              ; Adjust msg size
51  06  C0              ADDL    #TR3$C_HSZ_DATA,R1              ; Skip over Transport header
    3E  11              BRB     40$                             ; Join common code
                10$:    ;
                        ;
                        ;   Phase IV Router/Endnode
                        ;
                        ;
                        ASSUME  TR4$V_RTFLG_RTS  EQ  TR3$V_RTFLG_RTS
```

```
                                0215   609          ASSUME  TR4$V_RTFLG_RQR  EQ  TR3$V_RTFLG_RQR
                                0215   610
     39 22 A8    0A    E1       0215   611          BBC     #LPD$V_BC,LPD$W_STS(R8),40$     ; If BC, NOT a broadcast circuit
      53    51    0F    C3       021A   612          SUBL3   #HSZ_DELTA,R1,R3                 ; Point to new header area
            57    0F    C0       021E   613          ADDL    #HSZ_DELTA,R7                   ; Adjust msg size
                                0221   614          ;
                                0221   615          ;
                                0221   616          ;    For Broadcast Circuits, always set the Intra-NI flag.  It will be
                                0221   617          ;    cleared by routers if they route this packet off the Ethernet.
                                0221   618          ;
                                0221   619          ;
        83    81    24    89    0221   620          BISB3   #TR4$M_RTFLG_INI!-               ; Set the Intra-NI flag
                                0225   621                    TR4$M_RTFLG_LNG,(R1)+,(R3)+  ; Set the long format flag
              54    81    B0    0225   622          MOVW    (R1)+,R4                        ; Get destination address
              51    61    B0    0228   623          MOVW    (R1),R1                         ; Get source node address
                    83    B4    022B   624          CLRW    (R3)+                           ; RESERVED D-AREA, D-SUBAREA
  83    000400AA    8F    D0    022D   625          MOVL    #TR4$C_HIORD,(R3)+              ; Store destination HIORD
              83    54    B0    0234   626          MOVW    R4,(R3)+                        ; Store destination address
                    83    B4    0237   627          CLRW    (R3)+                           ; RESERVED S-AREA, D-SUBAREA
  83    000400AA    8F    D0    0239   628          MOVL    #TR4$C_HIORD,(R3)+              ; Store source HIORD
              83    51    B0    0240   629          MOVW    R1,(R3)+                        ; Store source node address
                    83    D4    0243   630          CLRL    (R3)+                           ; Clear NL2, VISIT-CT, SERVICE-
                                0245   631                                                  ;  CLASS and PROTOCOL TYPE
        51    EB    A3    9E    0245   632          MOVAB   -TR4$C_HSZ_DATA(R3),R1          ; Get new header pointer
                    08    11    0249   633          BRB     40$                             ; Join common code
                                024B   634 30$:      ;
                                024B   635          ;
                                024B   636          ;    Phase III Router/Endnode
                                024B   637          ;
                                024B   638          ;
                                024B   639          ;        ***** THE FOLLOWING IS A REQUIREMENT OF THE ARCHITECTURE *****
                                024B   640          ;
                                024B   641          ;        There are no known DECnet implementations which can handle node
                                024B   642          ;        addresses from other areas. Therefore, for Phase III nodes we
                                024B   643          ;        will always reset the area field of the source node address.
                                024B   644          ;        There are checks in the route-thru code to prevent route-thru
                                024B   645          ;        nodes from sending to Phase III nodes from other areas.
                                024B   646          ;
                                024B   647          ;
01 A1    FC00FC00    8F    CA    024B   648          BICL    #TR4$M_ADDR_AREA!<<TR4$M_ADDR_AREA>@16>,1(R1)
                                0253   649 40$:      ;
                                0253   650          ;
                                0253   651          ;    Pad the message if required
                                0253   652          ;
                                0253   653          ;
              53    51    D0    0253   654          MOVL    R1,R3                           ; Copy start of message pointer
        03 22 A8    0D    E1    0256   655          BBC     #LPD$V_ALIGNW,LPD$W_STS(R8),60$ ; If BC no word alignment needed
              51    01    CA    025B   656          BICL    #1,R1                           ; Else, backup to word boundary
        03 22 A8    0E    E1    025E   657 60$:      BBC     #LPD$V_ALIGNQ,LPD$W_STS(R8),80$ ; If BC no quad alignment needed
              51    07    CA    0263   658          BICL    #7,R1                           ; Else, backup to quad boundary
              53    51    C2    0266   659 80$:      SUBL    R1,R3                           ; Calculate size of rounding
                    08    13    0269   660          BEQL    100$                            ; Branch if no pad required
              57    53    C0    026B   661          ADDL    R3,R7                           ; Increase size of transfer
        61    53    80    8F    026E   662          BISB3   #^X<80>,R3,(R1)                 ; Set bit to indicate pad count
                          05    0273   663 100$:     RSB                                     ; Done
                                0274   664
```

```
                                0274   666          .SBTTL  QRL$SETUP_X_IRP - Allocate, init, queue IRP
                                0274   667 ;+
                                0274   668 ;
                                0274   669 ;  INPUTS:      R8      LPD address
                                0274   670 ;               R6      LPE address
                                0274   671 ;               R2      RCB address
                                0274   672 ;
                                0274   673 ;  OUTPUTS:     R0      LBS if there are IRP's and the LPD is active
                                0274   674 ;
                                0274   675 ;               R4,R3,R1  are clobbered
                                0274   676 ;
                                0274   677 ;               All others are unchanged
                                0274   678 ;
                                0274   679 ;-
                                0274   680 QRL$SETUP_X_IRP:                                ; Allocate, init, queue IRP
                                0274   681
                         50     D4     0274   682          CLRL    R0                      ; Assume error
                                0276   683          ASSUME  LPD$V_ACTIVE  EQ  0             ;
                   1C 22 A8     E9     0276   684          BLBC    LPD$W_STS(R8),100$      ; If BC, inactive
              02   10 A6     91     027A   685 10$:     CMPB    LPE$B_IRP_CNT(R6),#2    ; Do we already have two ?
                      13     1E     027E   686          BGEQU   90$                     ; If GEQU yes, that's enough
                      15     10     0280   687          BSBB    SETUP_X_IRP             ; Do it
                   09 50     E9     0282   688          BLBC    R0,50$                  ; If LBC, allocation failure
                   10 A6     96     0285   689          INCB    LPE$B_IRP_CNT(R6)       ; Bump the count
           0C B6     63     0E     0288   690          INSQUE  (R3),@LPE$Q_IRP_FREE+4(R6) ; Queue the IRP
                             028C   691 ;&       INCW    RCB$W_TRANS(R2)         ; Account for IRP
                      EC     11     028C   692          BRB     10$                     ; Loop
                   10 A6     95     028E   693 50$:     TSTB    LPE$B_IRP_CNT(R6)       ; Any IRP's ?
                      03     13     0291   694          BEQL    100$                    ; If EQL, return error
                50    01     D0     0293   695 90$:     MOVL    #1,R0                   ; Indicate success
                             05     0296   696 100$:     RSB                             ; Done
                                    0297   697
                                    0297   698
                                    0297   699 SETUP_X_IRP:
                                24  BB     0297   700          PUSHR   #^M<R2,R5>              ; Save regs
                          51    C4 8F     9A     0299   701          MOVZBL  #IRP$C_LENGTH,R1        ; Setup IRP size
                    00000000'GF     16     029D   702          JSB     G^EXE$ALONONPAGED       ; Get the block
                             3C 50  E9     02A3   703          BLBC    R0,200$                 ; Br on error
                                    02A6   704          ;
                                    02A6   705          ;
                                    02A6   706          ;    Zero the IRP and fill in selected fields
                                    02A6   707          ;
                                    02A6   708          ;
                                52  DD     02A6   709          PUSHL   R2                      ; Save block address
          62    00C4 8F     00    6E   00  2C     02A8   710          MOVC5   #0,(SP),#0,#IRP$C_LENGTH,(R2)   ; Zero entire block
                                53 8ED0     02B0   711          POPL    R3                      ; Setup IRP pointer
                                    02B3   712                                          ;
                          54    08 A3     9E     02B3   713          MOVAB   IRP$W_SIZE(R3),R4       ; Advance to size field
                                    02B7   714
                                    02B7   715          ASSUME  IRP$B_TYPE  EQ  2+IRP$W_SIZE
                                    02B7   716          ASSUME  IRP$B_RMOD  EQ  1+IRP$B_TYPE
                                    02B7   717
                          84    00C4 8F     B0     02B7   718          MOVW    #IRP$C_LENGTH,(R4)+     ; Enter size for deallocation
                                84    0A     9B     02BC   719          MOVZBW  S^#DYN$C_IRP,(R4)+      ; Enter buffer type and zero
                                    02BF   720                                          ; RMOD to indicate "kernel"
                                    02BF   721
                                    02BF   722          ASSUME  IRP$L_PID       EQ  1+IRP$B_RMOD
```

```
                     02BF    723          ASSUME  IRP$L_AST       EQ  4+IRP$L_PID
                     02BF    724          ASSUME  IRP$L_ASTPRM    EQ  4+IRP$L_AST
                     02BF    725          ASSUME  IRP$L_WIND      EQ  4+IRP$L_ASTPRM
                     02BF    726          ASSUME  IRP$L_UCB       EQ  4+IRP$L_WIND
                     02BF    727          ASSUME  LPD$L_UCB       EQ  4+LPD$L_WIND
                     02BF    728
 84   FE10 CF   9E   02BF    729          MOVAB   QRL$RTRN_X_IO,(R4)+              ; Enter return address into PID
   84   20 A8   3C   02C4    730          MOVZWL  LPD$W_PTR(R8),(R4)+             ; Enter LPD i.d. into AST
     84   6E   D0   02C8    731          MOVL    (SP),(R4)+                      ; Enter RCB ptr into ASTPRM
   84   0C A8   7D   02CB    732          MOVQ    LPD$L_WIND(R8),(R4)+            ; Enter WIND and UCB ptrs
                     02CF    733
                     02CF    734          ASSUME  IRP$W_FUNC      EQ  4+IRP$L_UCB
                     02CF    735          ASSUME  IRP$B_EFN       EQ  2+IRP$W_FUNC
                     02CF    736          ASSUME  IRP$B_PRI       EQ  1+IRP$B_EFN
                     02CF    737          ASSUME  IRP$L_IOSB      EQ  1+IRP$B_PRI
                     02CF    738          ASSUME  IRP$W_CHAN      EQ  4+IRP$L_IOSB
                     02CF    739          ASSUME  IRP$W_STS       EQ  2+IRP$W_CHAN
                     02CF    740
         84   7C     02CF    741          CLRQ    (R4)+                           ; Clear FUNC,EFN,PRI,IOSB
   84   14 A8   AE   02D1    742          MNEGW   LPD$W_CHAN(R8),(R4)+            ; Enter CHAN
         64   B4     02D5    743          CLRW    (R4)                            ; Setup STS for direct I/O write
03 22 A8   05   E1   02D7    744          BBC     #LPD$V_XBF,LPD$W_STS(R8),100$   ; If BC, writes are direct
      64   01   A8   02DC    745          BISW    #IRP$M_BUFIO,(R4)               ; Setup for buffered I/O
                     02DF    746                                                  ; and next reserved word
      50   01   D0   02DF    747 100$:    MOVL    #1,R0                           ; Indicate success
           24   BA   02E2    748 200$:    POPR    #^M<R2,R5>                      ; Recover regs
           05   02E4    749          RSB                                     ; Done
                     02E5    750
                     02E5    751
                     02E5    752
                     02E5    753 .END
```

```
$$_NSPMSG                       = 00000000
$$_TR3MSG                       = 00000000
$$_TR4MSG                       = 00000000
ACP$C_STA_F                     = 00000004
ACP$C_STA_H                     = 00000005
ACP$C_STA_I                     = 00000000
ACP$C_STA_N                     = 00000001
ACP$C_STA_R                     = 00000002
ACP$C_STA_S                     = 00000003
ADJ$B_PTYPE                     = 00000001
ADJ$C_PTY_AREA                  = 00000003
ADJ$C_PTY_PH3                   = 00000000
ADJ$C_PTY_PH3N                  = 00000001
ADJ$C_PTY_PH4                   = 00000004
ADJ$C_PTY_PH4N                  = 00000005
ADJ$W_PNA                       = 00000004
BUG$_NETNOSTATE                   ********    X   02
CA$_MEASURE                     = 00000001
CNF$_ADVANCE                    = 00000000
CNF$_QUIT                       = 00000002
CNF$_TAKE_CURR                  = 00000003
CNF$_TAKE_PREV                  = 00000001
COM$DRVDEALMEM                    ********    X   02
CXB$B_R_AREA                      00000039
CXB$B_R_FLG                       00000038
CXB$B_R_NSPTYP                    00000039
CXB$B_X_NSPTYP                    0000004E
CXB$C_DLL                       = 00000020
CXB$C_HEADER                    = 00000048
CXB$C_R_LENGTH                  = 0000003C
CXB$L_R_MSG                       0000002C
CXB$L_R_RCB                       00000028
CXB$T_DLL                       = 00000028
CXB$T_X_DATA                      00000057
CXB$T_X_XPORT                     00000048
CXB$W_R_ADJ                       0000003A
CXB$W_R_BCNT                      00000030
CXB$W_R_DSTNOD                    00000034
CXB$W_R_NSPSEQ                    0000003A
CXB$W_R_PATH                      00000032
CXB$W_R_SRCNOD                    00000036
CXB$W_X_NSPACK                    00000053
CXB$W_X_NSPLOC                    00000051
CXB$W_X_NSPREM                    0000004F
CXB$W_X_NSPSEQ                    00000055
DYN$C_IRP                       = 0000000A
EXE$ALONONPAGED                   ********    X   02
EXE$ALTQUEPKT                     ********    X   02
FINISH_XMT                        00000078  R     02
FKB$L_FPC                       = 0000000C
FKB$L_FR3                       = 00000010
FKB$L_FR4                       = 00000014
HSZ_DELTA                       = 0000000F
IO$_WRITELBLK                   = 00000020
IRP$B_EFN                       = 00000022
IRP$B_PRI                       = 00000023
IRP$B_RMOD                      = 0000000B
IRP$B_TYPE                      = 0000000A
IRP$C_LENGTH                    = 000000C4
IRP$L_AST                       = 00000010
IRP$L_ASTPRM                    = 00000014
IRP$L_BCNT                      = 00000032
IRP$L_IOSB                      = 00000024
IRP$L_IOST1                     = 00000038
IRP$L_PID                       = 0000000C
IRP$L_SAVD_RTN                  = 00000078
IRP$L_SVAPTE                    = 0000002C
IRP$L_UCB                       = 0000001C
IRP$L_WIND                      = 00000018
IRP$M_BUFIO                     = 00000001
IRP$Q_STATION                   = 00000040
IRP$W_BCNT                      = 00000032
IRP$W_BOFF                      = 00000030
IRP$W_CHAN                      = 00000028
IRP$W_FUNC                      = 00000020
IRP$W_SIZE                      = 00000008
IRP$W_STS                       = 0000002A
LPD$L_CNT_DPS                   = 00000042
LPD$L_UCB                       = 00000010
LPD$L_WIND                      = 0000000C
LPD$V_ACTIVE                    = 00000000
LPD$V_ALIGNQ                    = 0000000E
LPD$V_ALIGNW                    = 0000000D
LPD$V_BC                        = 0000000A
LPD$V_X25                       = 00000007
LPD$V_XBF                       = 00000005
LPD$W_CHAN                      = 00000014
LPD$W_PTH                       = 00000020
LPD$W_STS                       = 00000022
LPE$B_IRP_CNT                   = 00000010
LPE$C_LENGTH                    = 00000014
LPE$Q_IRP_FREE                  = 00000008
LPE$Q_IRP_WAIT                  = 00000000
LSB                             = 00000000
LSB$B_R_CXBCNT                  = 00000028
LSB$B_R_CXBQUO                  = 00000029
LSB$B_SPARE                     = 0000002A
LSB$B_STS                       = 0000002B
LSB$B_X_ADJ                     = 0000000B
LSB$B_X_CXBACT                  = 0000000D
LSB$B_X_CXBCNT                  = 0000000F
LSB$B_X_CXBQUO                  = 0000000E
LSB$B_X_PKTWND                  = 0000000C
LSB$B_X_REQ                     = 0000000A
LSB$L_CROSS                     = 0000002C
LSB$L_R_CXB                     = 00000020
LSB$L_R_IRP                     = 0000001C
LSB$L_X_CXB                     = 00000018
LSB$L_X_IRP                     = 00000014
LSB$L_X_PND                     = 00000010
LSB$M_BOM                       = 00000020
LSB$M_EOM                       = 00000040
LSB$M_LI                        = 00000001
LSB$S_LSB                       = 00000030
```

```
LSB$S_SPARE                     =  00000004
LSB$S_STS                       =  00000001
LSB$V_BOM                       =  00000005
LSB$V_EOM                       =  00000006
LSB$V_LI                        =  00000000
LSB$V_SPARE                     =  00000001
LSB$W_HAA                       =  00000008
LSB$W_HAR                       =  00000006
LSB$W_HAX                       =  00000026
LSB$W_HNR                       =  00000024
LSB$W_HXS                       =  00000004
LSB$W_LNX                       =  00000002
LSB$W_LUX                       =  00000000
MAX_C_LPE                       =  00000008
MMG$GL_SPTBASE                     ********    X    02
NET$ALONPGD_Z                      ********    X    02
NET$C_ACT_TIMER                 =  0000001E
NET$C_EFN_ASYN                  =  00000002
NET$C_EFN_WAIT                  =  00000001
NET$C_IPL                       =  00000008
NET$C_MAXACCFLD                 =  00000027
NET$C_MAXLINNAM                 =  0000000F
NET$C_MAXLNK                    =  000003FF
NET$C_MAXNODNAM                 =  00000006
NET$C_MAXOBJNAM                 =  0000000C
NET$C_MAX_AREAS                 =  0000003F
NET$C_MAX_LINES                 =  00000040
NET$C_MAX_NCB                   =  0000006E
NET$C_MAX_NODES                 =  000003FF
NET$C_MAX_OBJ                   =  000000FF
NET$C_MAX_WQE                   =  00000014
NET$C_MINBUFSIZ                 =  000000C0
NET$C_TID_ACT                   =  00000003
NET$C_TID_RUS                   =  00000001
NET$C_TID_XRT                   =  00000002
NET$C_TRCTL_CEL                 =  00000002
NET$C_TRCTL_OVR                 =  00000005
NET$C_UTLBUFSIZ                 =  00001000
NET$M_MAXLNKMSK                 =  000003FF
NET$WCB                            ********    X    02
NSP$$S_QUAL_ACK                 =  00000000
NSP$$S_QUAL_ALTFLW              =  00000000
NSP$$S_QUAL_DATA                =  00000000
NSP$$S_QUAL_FLW                 =  00000000
NSP$$S_QUAL_INF                 =  00000000
NSP$$S_QUAL_MSG                 =  00000000
NSP$$S_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_MAXRDR                    =  00000009
NSP$C_MAX_DELAY                 =  00000014
NSP$C_MAX_R_CXB                 =  00000007
NSP$C_MAX_XPW                   =  00000007
NSP$C_MSG_CA                    =  00000024
NSP$C_MSG_CC                    =  00000028
NSP$C_MSG_CI                    =  00000018
NSP$C_MSG_DATA                  =  00000000
NSP$C_MSG_DC                    =  00000048
NSP$C_MSG_DI                    =  00000038
NSP$C_MSG_DTACK                 =  00000004
NSP$C_MSG_INT                   =  00000030
NSP$C_MSG_LIACK                 =  00000014
NSP$C_MSG_LS                    =  00000010
NSP$C_SRV_MFC                   =  00000002
NSP$C_SRV_NFC                   =  00000000
NSP$C_SRV_REQ                   =  00000001
NSP$C_SRV_SFC                   =  00000001
NSP$M_ACK_NAK                   =  00001000
NSP$M_ACK_NUM                   =  00000FFF
NSP$M_ACK_VALID                 =  00008000
NSP$M_DATA_BOM                  =  00000020
NSP$M_DATA_EOM                  =  00000040
NSP$M_DATA_OVFW                 =  00000080
NSP$M_FLW_CHAN                  =  0000000C
NSP$M_FLW_DRV                   =  000000F0
NSP$M_FLW_INT                   =  00000020
NSP$M_FLW_INUSE                 =  00000010
NSP$M_FLW_LISUB                 =  00000004
NSP$M_FLW_MODE                  =  00000003
NSP$M_FLW_SP1                   =  00000008
NSP$M_FLW_SP2                   =  00000040
NSP$M_FLW_SP3                   =  00000080
NSP$M_FLW_XOFF                  =  00000001
NSP$M_FLW_XON                   =  00000002
NSP$M_INF_VER                   =  00000003
NSP$M_MSG_INT                   =  00000020
NSP$M_MSG_LI                    =  00000010
NSP$M_SRV_01                    =  00000003
NSP$M_SRV_EXT                   =  00000080
NSP$M_SRV_FLW                   =  0000000C
NSP$M_SRV_REQ                   =  000000F3
NSP$M_SRV_SP1                   =  00000070
NSP$R_QUAL                      =  00000000
NSP$S_ACK_NUM                   =  0000000C
NSP$S_ACK_SP2                   =  00000002
NSP$S_DATA_SP                   =  00000005
NSP$S_FLW_CHAN                  =  00000002
```

```
NSP$S_FLW_DRV                   =  00000004
NSP$S_FLW_MODE                  =  00000002
NSP$S_INF_VER                   =  00000002
NSP$S_MSG_SP1                   =  00000004
NSP$S_NSPMSG                    =  00000005
NSP$S_QUAL                      =  00000005
NSP$S_QUAL_ACK                  =  00000002
NSP$S_QUAL_ALTFLW               =  00000001
NSP$S_QUAL_DATA                 =  00000001
NSP$S_QUAL_FLW                  =  00000001
NSP$S_QUAL_INF                  =  00000001
NSP$S_QUAL_MSG                  =  00000005
NSP$S_QUAL_SRV                  =  00000001
NSP$S_SRV_01                    =  00000002
NSP$S_SRV_FLW                   =  00000002
NSP$S_SRV_SP1                   =  00000003
NSP$V_ACK_NAK                   =  0000000C
NSP$V_ACK_NUM                   =  00000000
NSP$V_ACK_SP2                   =  0000000D
NSP$V_ACK_VALID                 =  0000000F
NSP$V_DATA_BOM                  =  00000005
NSP$V_DATA_EOM                  =  00000006
NSP$V_DATA_OVFW                 =  00000007
NSP$V_DATA_SP                   =  00000000
NSP$V_FLW_CHAN                  =  00000002
NSP$V_FLW_DRV                   =  00000004
NSP$V_FLW_INT                   =  00000005
NSP$V_FLW_INUSE                 =  00000004
NSP$V_FLW_LISUB                 =  00000002
NSP$V_FLW_MODE                  =  00000000
NSP$V_FLW_SP1                   =  00000003
NSP$V_FLW_SP2                   =  00000006
NSP$V_FLW_SP3                   =  00000007
NSP$V_FLW_XOFF                  =  00000000
NSP$V_FLW_XON                   =  00000001
NSP$V_INF_VER                   =  00000000
NSP$V_MSG_INT                   =  00000005
NSP$V_MSG_LI                    =  00000004
NSP$V_MSG_SP1                   =  00000000
NSP$V_SRV_01                    =  00000000
NSP$V_SRV_EXT                   =  00000007
NSP$V_SRV_FLW                   =  00000002
NSP$V_SRV_SP1                   =  00000004
NSP$W_DSTLNK                    =  00000001
NSP$W_SRCLNK                    =  00000003
PMS$GL_DEPLOCPK                    ********    X   02
PMS$GL_RCVBUFFL                    ********    X   02
PR$_IPL                            ********    X   02
QRL$DENY                           00000035  R     02
QRL$GRANT                          00000041  R     02
QRL$RTRN_X_IO                      000000D3  R     02
QRL$RTRN_X_IRP                     0000012D  R     02
QRL$SETUP_CHAN                     0000015B  RG    02
QRL$SETUP_RTHDR                    000001F4  R     02
QRL$SETUP_X_IRP                    00000274  R     02
QRL$SOLICIT                        00000000  RG    02
QRL$XMT_ABORTED                    00000129  R     02
QUICK_SOL                          0000001F  R     02
RCB$B_ETY                       =  0000008A
RCB$L_PTR_ADJ                   =  0000002C
RCB$L_PTR_LPD                   =  00000028
SETUP_X_IRP                        00000297  R     02
SIZ...                          =  00000001
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$C_PRI_RTHRU                  =  0000001F
TR$GET_ADJ                         ********    X   02
TR3$$$_QUAL_MSG                 =  00000000
TR3$$$_QUAL_RTFLG               =  00000000
TR3$C_RSZ_DATA                  =  00000006
TR3$C_MSG_DATA                  =  00000002
TR3$C_MSG_HELLO                 =  00000005
TR3$C_MSG_INIT                  =  00000001
TR3$C_MSG_NOP2                  =  00000008
TR3$C_MSG_ROUT                  =  00000007
TR3$C_MSG_STR2                  =  00000058
TR3$C_MSG_VERF                  =  00000003
TR3$M_MSG_CTL                   =  00000001
TR3$M_MSG_RTH                   =  00000002
TR3$M_RTFLG_PH2                 =  00000040
TR3$M_RTFLG_RQR                 =  00000008
TR3$M_RTFLG_RTS                 =  00000010
TR3$R_QUAL                      =  00000000
TR3$S_QUAL                      =  00000001
TR3$S_QUAL_MSG                  =  00000001
TR3$S_QUAL_RTFLG                =  00000001
TR3$S_RTFLG_012                 =  00000003
TR3$S_TR3MSG                    =  00000001
TR3$V_MSG_CTL                   =  00000000
TR3$V_MSG_RTH                   =  00000001
TR3$V_RTFLG_012                 =  00000000
TR3$V_RTFLG_5                   =  00000005
TR3$V_RTFLG_7                   =  00000007
TR3$V_RTFLG_PH2                 =  00000006
TR3$V_RTFLG_RQR                 =  00000003
TR3$V_RTFLG_RTS                 =  00000004
TR4$$$_QUAL_ADDR                =  00000000
TR4$$$_QUAL_RTFLG               =  00000000
TR4$$$_QUAL_SCLASS              =  00000000
TR4$C_BCE_MID1                  =  040000AB
TR4$C_BCE_MID2                  =  00000000
TR4$C_BCR_MID1                  =  030000AB
TR4$C_BCR_MID2                  =  00000000
TR4$C_BCT3MULT                  =  00000008
TR4$C_END_NODE                  =  00000003
TR4$C_HIORD                     =  000400AA
TR4$C_HSZ_DATA                  =  00000015
TR4$C_MSG_BCEHEL                =  0000000D
```

```
TR4$C_MSG_BCRHEL                =  0000000B
TR4$C_MSG_LDATA                 =  00000006
TR4$C_MSG_RDATA                 =  00000002
TR4$C_PRO_TYPE                  =  00000360
TR4$C_RTR_LVL1                  =  00000002
TR4$C_RTR_LVL2                  =  00000001
TR4$C_T3MULT                    =  00000002
TR4$C_VER_HIB                   =  00000000
TR4$C_VER_LOWW                  =  00000002
TR4$M_ADDR_AREA                 =  0000FC00
TR4$M_ADDR_DEST                 =  000003FF
TR4$M_RTFLG_INI                 =  00000020
TR4$M_RTFLG_LNG                 =  00000004
TR4$M_RTFLG_RQR                 =  00000008
TR4$M_RTFLG_RTS                 =  00000010
TR4$R_QUAL                      =  00000000
TR4$S_ADDR_AREA                 =  00000006
TR4$S_ADDR_DEST                 =  0000000A
TR4$S_QUAL                      =  00000002
TR4$S_QUAL_ADDR                 =  00000002
TR4$S_QUAL_RTFLG                =  00000001
TR4$S_QUAL_SCLASS               =  00000001
TR4$S_RTFLG_01                  =  00000002
TR4$S_RTFLG_VER                 =  00000002
TR4$S_SCLASS_57                 =  00000003
TR4$S_TR4MSG                    =  00000002
TR4$V_ADDR_AREA                 =  0000000A
TR4$V_ADDR_DEST                 =  00000000
TR4$V_RTFLG_01                  =  00000000
TR4$V_RTFLG_INI                 =  00000005
TR4$V_RTFLG_LNG                 =  00000002
TR4$V_RTFLG_RQR                 =  00000003
TR4$V_RTFLG_RTS                 =  00000004
TR4$V_RTFLG_VER                 =  00000006
TR4$V_SCLASS_1                  =  00000001
TR4$V_SCLASS_57                 =  00000005
TR4$V_SCLASS_BC                 =  00000004
TR4$V_SCLASS_LS                 =  00000002
TR4$V_SCLASS_METR               =  00000000
TR4$V_SCLASS_SUBA               =  00000003
VA$M_BYTE                       =  000001FF
VA$S_VPN                        =  00000015
VA$V_VPN                        =  00000009
XWB                             =  00000000
XWB$B_ACCESS                    =  0000000B
XWB$B_DATA                      =  0000005B
XWB$B_FIPL                      =  0000001F
XWB$B_LOGIN                     =  000000CC
XWB$B_LPRNAM                    =  000000A4
XWB$B_PRO                       =  0000005A
XWB$B_RID                       =  0000006F
XWB$B_RPRNAM                    =  000000B8
XWB$B_SP3                       =  0000006E
XWB$B_STA                       =  0000001E
XWB$B_TYPE                      =  0000000A
XWB$B_X_FLW                     =  0000006C
XWB$B_X_FLWCNT                  =  0000006D
XWB$C_COMLNG                    =  000000A4
XWB$C_CONLNG                    =  00000112
XWB$C_DATA                      =  00000010
XWB$C_LOGIN                     =  00000040
XWB$C_LPRNAM                    =  00000014
XWB$C_NDC_LNG                   =  00000020
XWB$C_NUMSTA                    =  00000008
XWB$C_RID                       =  00000010
XWB$C_RPRNAM                    =  00000014
XWB$C_STA_CAR                   =  00000002
XWB$C_STA_CCS                   =  00000004
XWB$C_STA_CIR                   =  00000003
XWB$C_STA_CIS                   =  00000001
XWB$C_STA_CLO                   =  00000000
XWB$C_STA_DIR                   =  00000006
XWB$C_STA_DIS                   =  00000007
XWB$C_STA_RUN                   =  00000005
XWB$L_DEA_IRP                   =  00000104
XWB$L_FPC                       =  00000020
XWB$L_FR3                       =  00000024
XWB$L_FR4                       =  00000028
XWB$L_ICB                       =  0000010C
XWB$L_IRP_ACC                   =  00000080
XWB$L_LINK                      =  0000002C
XWB$L_ORGUCB                    =  00000010
XWB$L_PID                       =  00000034
XWB$L_VCB                       =  00000030
XWB$L_WLBL                      =  00000004
XWB$L_WLFL                      =  00000000
XWB$M_FLG_BREAK                 =  00000001
XWB$M_FLG_CLO                   =  00000200
XWB$M_FLG_IAVL                  =  00001000
XWB$M_FLG_SCD                   =  00000100
XWB$M_FLG_SDACK                 =  00000008
XWB$M_FLG_SDFL                  =  00004000
XWB$M_FLG_SDT                   =  00000080
XWB$M_FLG_SIACK                 =  00000004
XWB$M_FLG_SIFL                  =  00002000
XWB$M_FLG_SLI                   =  00000010
XWB$M_FLG_TBPR                  =  00000800
XWB$M_FLG_WBP                   =  00000040
XWB$M_FLG_WBUF                  =  00000002
XWB$M_FLG_WDAT                  =  00000400
XWB$M_FLG_WHGL                  =  00000020
XWB$M_PRO_CCA                   =  00000008
XWB$M_PRO_NAR                   =  00000010
XWB$M_PRO_NFC                   =  00000001
XWB$M_PRO_PH2                   =  00000004
XWB$M_PRO_SFC                   =  00000002
XWB$M_STS_ASTPND                =  00000400
XWB$M_STS_ASTREQ                =  00000800
XWB$M_STS_CON                   =  00000010
XWB$M_STS_DIS                   =  00000008
XWB$M_STS_DTNAK                 =  00001000
XWB$M_STS_LINAK                 =  00000200
XWB$M_STS_NDC                   =  00001000
XWB$M_STS_OVF                   =  00000080
```

```
XWB$M_STS_RBP                   = 00000040
XWB$M_STS_SOL                   = 00000004
XWB$M_STS_TID                   = 00000001
XWB$M_STS_TLI                   = 00000002
XWB$M_STS_TMO                   = 00000020
XWB$Q_FORK                      = 00000014
XWB$Q_FREE_CXB                    00000118
XWB$R_CON_BLK                   = 000000A4
XWB$R_RUN_BLK                   = 000000A4
XWB$S_                          = 00000006
XWB$S_COMLNG                    = 0000006E
XWB$S_CON_BLK                   = 0000006E
XWB$S_DATA                      = 00000010
XWB$S_DT                        = 00000030
XWB$S_FLG                       = 00000002
XWB$S_FORK                      = 00000008
XWB$S_FREE_CXB                  = 00000008
XWB$S_LI                        = 00000030
XWB$S_LOGIN                     = 0000003F
XWB$S_LPRNAM                    = 00000013
XWB$S_NDC                       = 00000020
XWB$S_PRO                       = 00000001
XWB$S_RID                       = 00000010
XWB$S_RPRNAM                    = 00000013
XWB$S_RUN_BLK                   = 00000064
XWB$S_STS                       = 00000002
XWB$S_XWB                       = 00000120
XWB$T_                          = 00000112
XWB$T_DATA                      = 0000005C
XWB$T_DT                        = 000000A4
XWB$T_LI                        = 000000D4
XWB$T_LOGIN                     = 000000CD
XWB$T_LPRNAM                    = 000000A5
XWB$T_RID                       = 00000070
XWB$T_RPRNAM                    = 000000B9
XWB$V_FLG_BREAK                 = 00000000
XWB$V_FLG_CLO                   = 00000009
XWB$V_FLG_IAVL                  = 0000000C
XWB$V_FLG_SCD                   = 00000008
XWB$V_FLG_SDACK                 = 00000003
XWB$V_FLG_SDFL                  = 0000000E
XWB$V_FLG_SDT                   = 00000007
XWB$V_FLG_SIACK                 = 00000002
XWB$V_FLG_SIFL                  = 0000000D
XWB$V_FLG_SLI                   = 00000004
XWB$V_FLG_TBPR                  = 0000000B
XWB$V_FLG_WBP                   = 00000006
XWB$V_FLG_WBUF                  = 00000001
XWB$V_FLG_WDAT                  = 0000000A
XWB$V_FLG_WHGL                  = 00000005
XWB$V_PRO_CCA                   = 00000003
XWB$V_PRO_NAR                   = 00000004
XWB$V_PRO_NFC                   = 00000000
XWB$V_PRO_PH2                   = 00000002
XWB$V_PRO_SFC                   = 00000001
XWB$V_STS_ASTPND                = 0000000A
XWB$V_STS_ASTREQ                = 0000000B
XWB$V_STS_CON                   = 00000004
XWB$V_STS_DIS                   = 00000003
XWB$V_STS_DTNAK                 = 00000008
XWB$V_STS_LINAK                 = 00000009
XWB$V_STS_NDC                   = 0000000C
XWB$V_STS_OVF                   = 00000007
XWB$V_STS_RBP                   = 00000006
XWB$V_STS_SOL                   = 00000002
XWB$V_STS_TID                   = 00000000
XWB$V_STS_TLI                   = 00000001
XWB$V_STS_TMO                   = 00000005
XWB$W_CI_PATH                   = 00000110
XWB$W_DECAY                     = 0000004E
XWB$W_DLY_FACT                  = 00000056
XWB$W_DLY_WGHT                  = 00000058
XWB$W_ELAPSE                    = 0000004A
XWB$W_FLG                       = 0000001C
XWB$W_LOCLNK                    = 0000003E
XWB$W_LOCSIZ                    = 00000040
XWB$W_PATH                      = 00000038
XWB$W_PROGRESS                  = 00000052
XWB$W_REFCNT                    = 0000000C
XWB$W_REMLNK                    = 0000003C
XWB$W_REMNOD                    = 0000003A
XWB$W_REMSIZ                    = 00000042
XWB$W_RETRAN                    = 00000054
XWB$W_R_REASON                  = 00000044
XWB$W_SIZE                      = 00000008
XWB$W_STS                       = 0000000E
XWB$W_TIMER                     = 00000050
XWB$W_TIM_ID                    = 00000048
XWB$W_TIM_INACT                 = 0000004C
XWB$W_X_REASON                  = 00000046
XWB$Z_NDC                       = 00000084
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000057 (87.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	000002E5 (741.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.09	00:00:01.29
Command processing	152	00:00:01.04	00:00:05.78
Pass 1	524	00:00:20.74	00:00:55.10
Symbol table sort	0	00:00:03.61	00:00:07.15
Pass 2	142	00:00:03.86	00:00:09.29
Symbol table output	65	00:00:00.47	00:00:01.41
Psect synopsis output	4	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	922	00:00:29.84	00:01:20.05

The working set limit was 2000 pages.
115222 bytes (226 pages) of virtual memory were used to buffer the intermediate code.
There were 120 pages of symbol table space allocated to hold 2276 non-local and 41 local symbols.
753 source lines were read in Pass 1, producing 15 object records in Pass 2.
57 pages of virtual memory were used to define 39 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]NMALIBRY.MLB;1	0
_255DUA28:[SHRLIB]EVCDEF.MLB;1	0
_255DUA28:[NETACP.OBJ]NETDRV.MLB;1	3
_255DUA28:[NETACP.OBJ]NET.MLB;1	7
_255DUA28:[SYS.OBJ]LIB.MLB;1	11
_255DUA28:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	29

2504 GETS were required to define 29 macros.

There were no errors, warnings or information messages.

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

NETDRVQRL
LIS

NETDRVSES
LIS

NETDRVNSP
LIS