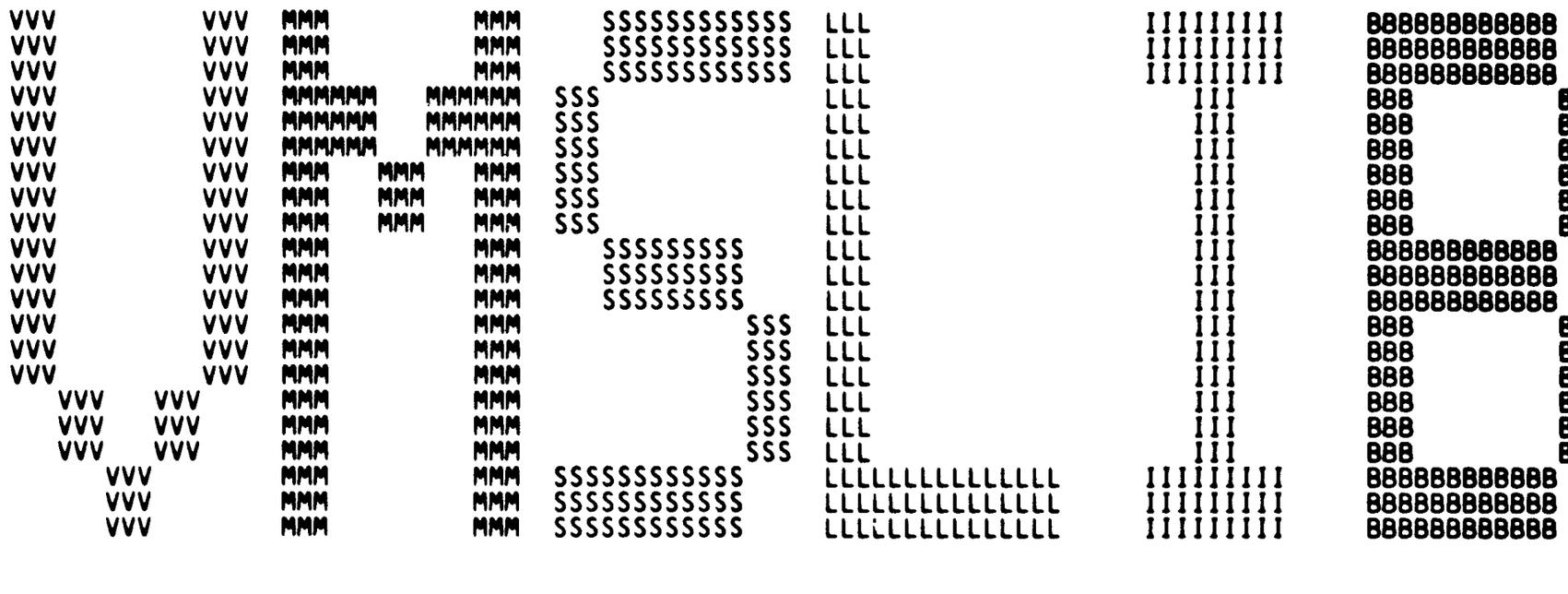
VVV VVV MMM MMM SSSSSSSSSSS LLL IIIIIIIII BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSSSS LLL IIIIIIIII BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSSSS LLL IIIIIIIII BBBBBBBBBBB
VVV VVV MMMMMM MMMMMM SSS LLL III BBB BBB
VVV VVV MMMMMM MMMMMM SSS LLL III BBB BBB
VVV VVV MMMMMM MMMMMM SSS LLL III BBB BBB
VVV VVV MMM MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSSSSSSSS LLL III BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSS LLL III BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSS LLL III BBBBBBBBBBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSSSSSSSSSS LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSSSS LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBB
VVV MMM MMM SSSSSSSSSSS LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBB

FILEID**LIBNETCON

```
LL              IIIIII    BBBBBBBB    NN      NN  EEEEEEEEEE  TTTTTTTTTT    CCCCCCCC    000000    NN      NN
LL              IIIIII    BBBBBBBB    NN      NN  EEEEEEEEEE  TTTTTTTTTT    CCCCCCCC    000000    NN      NN
LL                II      BB      BB  NN      NN  EE              TT      CC          00      00  NN      NN
LL                II      BB      BB  NN      NN  EE              TT      CC          00      00  NN      NN
LL                II      BB      BB  NNNN    NN  EE              TT      CC          00      00  NNNN    NN
LL                II      BB      BB  NNNN    NN  EE              TT      CC          00      00  NNNN    NN
LL                II      BBBBBBBB    NN  NN  NN  EEEEEEEE        TT      CC          00      00  NN  NN  NN
LL                II      BBBBBBBB    NN  NN  NN  EEEEEEEE        TT      CC          00      00  NN  NN  NN
LL                II      BB      BB  NN    NNNN  EE              TT      CC          00      00  NN    NNNN
LL                II      BB      BB  NN    NNNN  EE              TT      CC          00      00  NN    NNNN
LL                II      BB      BB  NN      NN  EE              TT      CC          00      00  NN      NN      ....
LL                II      BB      BB  NN      NN  EE              TT      CC          00      00  NN      NN      ....
LLLLLLLLLL      IIIIII    BBBBBBBB    NN      NN  EEEEEEEEEE      TT        CCCCCCCC    000000    NN      NN      ....
LLLLLLLLLL      IIIIII    BBBBBBBB    NN      NN  EEEEEEEEEE      TT        CCCCCCCC    000000    NN      NN      ....

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

Table of contents

```
0000     1 .TITLE  UNS$NET_CONNECT Make a (pass through) DECnet connection
0000     2 .IDENT  'V04-000'
0000     3
0000     4 ;
0000     5 ;**********************************************************************************
0000     6 ;*                                                                                *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                       *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                        *
0000     9 ;*  ALL RIGHTS RESERVED.                                                          *
0000    10 ;*                                                                                *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED         *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE        *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER         *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY         *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY        *
0000    16 ;*  TRANSFERRED.                                                                  *
0000    17 ;*                                                                                *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE         *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT        *
0000    20 ;*  CORPORATION.                                                                  *
0000    21 ;*                                                                                *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS         *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                       *
0000    24 ;*                                                                                *
0000    25 ;*                                                                                *
0000    26 ;**********************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ; FACILITY: DECnet library support routines
0000    31 ;
0000    32 ; ABSTRACT:
0000    33 ;
0000    34 ;       Makes a DECnet logical link connection, possibly passing
0000    35 ;       through many nodes to get to some non-adjacent node.
0000    36 ;
0000    37 ; ENVIRONMENT: VMS - USER MODE
0000    38 ;
0000    39 ; AUTHOR: Mark Bramhall           CREATION DATE: 25-Feb-1980
0000    40 ;
0000    41 ; MODIFICATION HISTORY:
0000    42 ;
0000    43 ;       V03-004         DAS0001         David Solomon           14-Apr-1984
0000    44 ;               Remove reference to unused RMS NOP bit, NAM$M_ROD.
0000    45 ;
0000    46 ;       V03-003         MHB0098         Mark Bramhall           10-Jan-1984
0000    47 ;               Handle lack of NETMBX privilege correctly.
0000    48 ;
0000    49 ;       V03-002         MHB0095         Mark Bramhall           25-Mar-1983
0000    50 ;               Preserve optional data with NAM$M_ROD in NAM$B_NOP.
0000    51 ;
0000    52 ;       V03-001         MHB0083         Mark Bramhall           14-Jan-1983
0000    53 ;               Use $PARSE to parse the "device name" specification.
0000    54 ;
0000    55 ;       V02-002         MHB0077         Mark Bramhall            9-Jun-1981
0000    56 ;               Added data message buffer size for "pass through" responders.
0000    57 ;
```

```
                     0000    58 ;--
```

```
0000     60 .SBTTL  UNS$NET_CONNECT Make a (pass through) DECnet connection
0000     61
0000     62 ;+
0000     63 ; UNS$NET_CONNECT - Make a (pass through) DECnet connection.
0000     64 ;
0000     65 ;       Makes a DECnet logical link connection, possibly using the
0000     66 ;       "pass through" protocol to get to a non-adjacent node.  The
0000     67 ;       connection can be made in either "transparent" mode (no mailbox
0000     68 ;       channel word argument) or "non-transparent" mode (mailbox channel
0000     69 ;       specified).  Note that for non-adjacent nodes (i.e., when the
0000     70 ;       "pass through" protocol is in use), the link mailbox is connected
0000     71 ;       to the adjacent node's "pass through" object type responder.
0000     72 ;       While most "pass through" object type responders will forward
0000     73 ;       interrupt messages in both directions, other DECnet generated
0000     74 ;       messages (especially the connection confirm/reject) are for the
0000     75 ;       first hop only.
0000     76 ;
0000     77 ;       Additionally, optional arguments allow for:
0000     78 ;
0000     79 ;       1) Obtaining the final node path used.
0000     80 ;
0000     81 ;       2) Obtaining the Access Control String (ACS), if any, specified
0000     82 ;               for the final target node.  If the final ACS is asked for,
0000     83 ;               it is stripped from the actual connection request, thus
0000     84 ;               applying the default ACS for the target node connection.
0000     85 ;               [Note: All intermediate connections using the "pass through"
0000     86 ;               protocol strip any user specified ACS and always use the
0000     87 ;               default ACS.]
0000     88 ;
0000     89 ;       3) If any informational, warning, and/or failure messages are
0000     90 ;               generated by the "pass through" protocol, they can also be
0000     91 ;               obtained.  As each message is generated, a user supplied
0000     92 ;               routine is called with a $PUTMSG message argument vector.
0000     93 ;               The status code used is SHR$_TEXT (shared message of simply
0000     94 ;               text) with the severity level set according to the message
0000     95 ;               type and with the facility code set to "LIB".
0000     96 ;
0000     97 ;       4) Passing a data message buffer size to any "pass through" object
0000     98 ;               responders.
0000     99 ;
0000    100 ;       Call format:
0000    101 ;
0000    102 ;       ret-status.wlc.v = UNS$NET_CONNECT ( device-name.rt.dx,
0000    103 ;                                            link-chan.ww.r,
0000    104 ;                                ** optional **   mailbox-chan.ww.r,
0000    105 ;                                ** optional **   path-list.wt.dx,
0000    106 ;                                ** optional **   final-acs.wt.dx,
0000    107 ;                                ** optional **   msg-routine.fz.rp,
0000    108 ;                                ** optional **   buffer-size.rw.r,
0000    109 ;    ** "pass through" object type only **   num-hops.rb.r )
0000    110 ;
0000    111 ;       Inputs:
0000    112 ;
0000    113 ;               00(AP) = Argument count (must be 2 or greater)
0000    114 ;               04(AP) = Address of "device name" descriptor
0000    115 ;               08(AP) = Address of word for assigned DECnet channel number
0000    116 ;               12(AP) = Address of word for assigned mailbox channel number
```

```
                        ;               16(AP) = Address of descriptor for resultant path list
                        ;               20(AP) = Address of descriptor for final Access Control String
                        ;               24(AP) = Address of routine to call with messages
                        ;               28(AP) = Address of "pass through" data buffer size word
                        ;       ***** This argument is for the "pass through" object type only *****
                        ;               32(AP) = Address of "number of hops" byte to pass if passing
                        ;       ***** This argument is for the "pass through" object type only *****
                        ;
                        ;       Outputs:
                        ;
                        ;               R0 = Status code
                        ;               Word @8(AP) = Assigned DECnet channel number
                        ;                             (zero if connection failure)
                        ;                             (non-zero if "pass through" protocol failure)
                        ;               Word @12(AP) = Assigned mailbox channel number
                        ;                             (zero if connection failure)
                        ;                             (non-zero if "pass through" protocol failure)
                        ;               Desc @16(AP) = Resultant path list descriptor
                        ;                             (attempted connection node name if failure)
                        ;               Desc @20(AP) = Final Access Control String descriptor
                        ;               Rout @24(AP) = Called with a $PUTMSG message argument vector
                        ;                             for "pass through" protocol generated messages
                        ;
                        ;       Assumptions:
                        ;
                        ;               The "device name" string is of the general format
                        ;
                        ;                       [node-list]dest-node["ACS"]::"object-type"
                        ;
                        ;               where <node-list> is (spaces are for clairity only!)
                        ;
                        ;                       node-1:: [ node-2:: [ node-3:: [ ... ] ] ]
                        ;
                        ;               where any node name (<dest-node> and/or <node-n>) can
                        ;               contain a leading underscore.
                        ;-
                        ;
                        ; Argument offsets
                        ;
00000004  0000          DEV_NAME_DESC   =       4               ; Offset to "device name" descriptor
00000008  0000          NET_CHAN_WORD   =       8               ; Offset to DECnet channel number word
0000000C  0000          MBX_CHAN_WORD   =      12               ; Offset to mailbox channel number word
00000010  0000          RSL_PATH_DESC   =      16               ; Offset to resu tant list descriptor
00000014  0000          FINL_ACS_DESC   =      20               ; Offset to final ACS descriptor
00000018  0000          MSG_ROUT_ADDR   =      24               ; Offset to message call routine
0000001C  0000          BUF_SIZE_WORD   =      28               ; Offset to data buffer size word
00000020  0000          NUM_HOPS_BYTE   =      32               ; Offset to "number of hops" byte
```

```
                                0000   165 ; Local definitions
                                0000   166
                                0000   167         $FABDEF                         ; FAB definitions
                                0000   168         $NAMDEF                         ; NAM definitions
                                0000   169         $SHRDEF                         ; Shared message definitions
                                0000   170         $STSDEF                         ; Status code definitions
                                0000   171
                    00000100    0000   172 IO_BUF_LEN      =       256             ; Size of I/O buffer for protocol I/O
                                0000   173
                                0000   174 ; Local offsets
                                0000   175
                    FFFFFFF8    0000   176 WAIT_EFN        =        -8             ; FP offset to our Event Flag Number
                    FFFFFFFC    0000   177 RET_STATUS      =        -4             ; FP offset to saved return status
                                0000   178
                                0000   179 ; .PSECT definition
                                0000   180
                          00000000  0000   181 .PSECT  _LIB$CODE       PIC, USR, CON, REL, LCL, SHR, EXE, RD, NOWRT, LONG
                                0000   182
                                0000   183 ; Local data
                                0000   184
                                0000   185 DOUBLE_COLON:                          ; Double colon for finding a node name
                             3A 3A  0000   186         .ASCII  /::/
                                0002   187
                                0002   188 PASS_OBJECT:                          ; Pass through protocol object type
                  2F 3D 33 32 31 22 3A 3A  0002   189         .ASCII  '::"123=/'
                                0000   190         .WORD   0
                                  00   000C   191 10$:     .BYTE   0
00'00'00'00'00'00'00'00'00'00'00'00'00'  000D   192         .BYTE   0 [16]
                        00'00'00'00'  001D
                                  22   001D   193         .ASCII  '"'
                          FFFFFFEE   001E   194 PASS_OBJECT_CNT =       10$-.
                          0000001C   001E   195 PASS_OBJECT_LEN =       .-PASS_OBJECT
                                001E   196
                                001E   197 NET_DEV_NAM:                         ; DECnet "device name"
                       3A 30 54 45 4E 5F   001E   198         .ASCII  /_NET0:/
                          00000006   0024   199 NET_DEV_NAM_LEN =       .-NET_DEV_NAM
                                0024   200
                                0024   201 CREMBX_BUFQUO:                       ; Link mailbox buffer quota (defaults)
                          00000000   0024   202         .LONG   0
                                0028   203
                                0028   204 CREMBX_MAXMSG:                       ; Link mailbox max msg len (DECnet=40)
                          00000028   0028   205         .LONG   40
                                002C   206
                                002C   207 .SHOW   MEB
                                002C   208
                                002C   209 STRING_FETCH:                        ; Fetch a string's length/address
                                0000   002C   210         .WORD   ^M<>
                                002E   211
6D   00000000'GF    9E          002E   212         MOVAB   G^LIB$SIG_TO_RET, (FP)  ; Return signals as status codes
00000000'GF    6C    FA         0035   213         CALLG   (AP), G^LIB$ANALYZE_SDESC ; Go fetch string's length/address
                       04       003C   214         RET                             ; Return.
                                003D   215
                                003D   216 STRING_COPY:                        ; Copy a string to caller's desc
                                0000   003D   217         .WORD   ^M<>
                                003F   218
6D   00000000'GF    9E          003F   219         MOVAB   G^LIB$SIG_TO_RET, (FP)  ; Return signals as status codes
00000000'GF    6C    FA         0046   220         CALLG   (AP), G^STR$COPY_DX     ; Go copy string to caller's desc
```

```
                                04  004D   221          RET                                 ; Return.
                                    004E   222
                              00FC  004E   223 .ENTRY  UNS$NET_CONNECT, ^M<R2,R3,R4,R5,R6,R7> ; Entry point and register mask
                                    0050   224
                             7E   7C  0050   225          CLRQ    -(SP)                       ; Make room for return status & EFN
                                  0052   226 ASSUME   RET_STATUS EQ -4
                                  0052   227 ASSUME   WAIT_EFN EQ -8
                          08 BC   B4  0052   228          CLRW    @NET_CHAN_WORD(AP)          ; Pre-clear DECnet channel number word
                             57   D4  0055   229          CLRL    R7                          ; Guess at no mailbox handling
                       03    6C   91  0057   230          CMPB    (AP), #MBX_CHAN_WORD/4      ; Do we have this argument?
                             0A   1F  005A   231          BLSSU   10$                         ; Nope
                          0C AC   D5  005C   232          TSTL    MBX_CHAN_WORD(AP)           ; Yep, but was it really supplied?
                             05   13  005F   233          BEQL    10$                         ; Not really there...
                          0C BC   B4  0061   234          CLRW    @MBX_CHAN_WORD(AP)          ; Pre-clear mailbox channel number word
                             57   D6  0064   235          INCL    R7                          ;  and say we're handling a mailbox
                                  0066   236 10$:                                            ; Continue
                                  0066   237
                                  0066   238 PARSE_STRING:                                   ; Parse the "device name" string
                             7E   7C  0066   239          CLRQ    -(SP)                       ; Form a cleared descriptor
                       56    6E   7E  0068   240          MOVAQ   (SP), R6                    ;  and get a pointer to it
                 5E    FEE4   CE   DE  006B   241          MOVAL   -<NAM$C_MAXRSS+PASS_OBJECT_LEN+- ; Form connect buffer
                                  0070   242                          <4-1>&^C<4-1>>(SP), SP ;  rounded up to a longword
                    04 A6    6E   9E  0070   243          MOVAB   (SP), 4(R6)                 ;   and save its address
                 5E    FF50   CE   DE  0074   244          MOVAL   -<FAB$C_BLN+NAM$C_BLN>(SP), SP ; Allocate a FAB and NAM
   6E    00B0 8F    00   6E   00   2C  0079   245          MOVC5   #0, (SP), #0, #FAB$C_BLN+NAM$C_BLN, (SP) ;  then clear them
                       52   6E   DE  0081   246          MOVAL   (SP), R2                    ; Address the FAB
                    53    50   AE   DE  0084   247          MOVAL   FAB$C_BLN(SP), R3           ;  and the NAM
                 62    5003 8F   B0  0088   248          MOVW    #<FAB$C_BLN@8>!FAB$_BID, FAB$B_BID(R2) ; Set up the FAB
                                  008D   249 ASSUME   FAB$B_BLN EQ FAB$B_BID+1
                    28 A2    63   DE  008D   250          MOVAL   (R3), FAB$L_NAM(R2)         ; Link the NAM to the FAB
                 63    6002 8F   B0  0091   251          MOVW    #<NAM$C_BLN@8>!NAM$C_BID, NAM$B_BID(R3) ; Set up the NAM
                                  0096   252 ASSUME   NAM$B_BLN EQ NAM$B_BID+1
                       08 A3    01   90  0096   253          MOVB    #NAM$M_PWD, NAM$B_NOP(R3) ; Keep password
                    0C A3    04 B6   9E  009A   254          MOVAB   @4(R6), NAM$L_ESA(R3)       ; Set expanded string address
                    0A A3    FF 8F   90  009F   255          MOVB    #NAM$C_MAXRSS, NAM$B_ESS(R3) ;  and expanded string length
                             2C A2   DF  00A4   256          PUSHAL  FAB$L_FNA(R2)               ; Arg #3 is the returned address
                             34 A2   3F  00A7   257          PUSHAW  FAB$B_FNS(R2)               ; Arg #2 is the returned length
                                  00AA   258 ASSUME   FAB$B_DNS EQ FAB$B_FNS+1
                          04 BC   7F  00AA   259          PUSHAQ  @DEV_NAME_DESC(AP)          ; Arg #1 is the "device name" desc
                 FF7A CF    03   FB  00AD   260          CALLS   #3, STRING_FETCH            ; Go fetch the string's length/address
                           020D   30  00B2   261          BSBW    LOWBIT_ELSE_DIE             ; Check the completion code
              50    00000000'8F   D0  00B5   262          MOVL    #RMS$_NOD, R0               ; Preset "Illegal node name" error
                          35 A2   95  00BC   263          TSTB    FAB$B_DNS(R2)               ; Was the "device name" too long?
                                  00BF   264 ASSUME   FAB$B_DNS EQ FAB$B_FNS+1
                             1B   12  00BF   265          BNEQ    10$                         ; Yes, go call it an error...
                                  00C1   266          $PARSE -                            ; Parse
                                  00C1   267                  FAB = (R2)                  ;  the "device name"
                             62   DF  00C1                   PUSHAL  (R2)
00000000'GF                  01   FB  00C3                   CALLS   #$$.TMP1,G^SYS$PARSE
00000000'8F                  50   D1  00CA   268          CMPL    R0, #RMS$_CHN               ; Did the channel assignment fail?
                             09   12  00D1   269          BNEQ    10$                         ; Nope
                       51    0C A2   D0  00D3   270          MOVL    FAB$L_STV(R2), R1           ; Yep, get the STV from the parse
                             03   13  00D7   271          BEQL    10$                         ; None
                          50    51   D0  00D9   272          MOVL    R1, R0                      ; One, use STV as failure code
                              01E3   30  00DC   273 10$:     BSBW    LOWBIT_ELSE_DIE             ; Check the completion code
                       54    38 A3   9A  00DF   274          MOVZBL  NAM$B_NODE(R3), R4          ; Form descriptor to the node name
                       55    40 B3   9E  00E3   275          MOVAB   @NAM$L_NODE(R3), R5         ;  part of string (to extract ACS)
```

```
                66   0B A3    9A  00E7    276          MOVZBL  NAM$B_ESL(R3), (R6)      ; Set expanded string's real length
                5E   04 B6    9E  00EB    277          MOVAB   @4(R6), SP               ; Flush (now) unused stack space
          50    00000000'8F   D0  00EF    278          MOVL    #RMS$_NOD, R0            ; Preset "Illegal node name" error
                   5E   55    D1  00F6    279          CMPL    R5, SP                   ; Is the node name the first thing?
                        E1    12  00F9    280          BNEQ    10$                      ; Nope, go call it an error...
                              00FB    281
                              00FB    282 GET_FINAL_ACS:                               ; Extract final ACS if needed
                05    6C    91  00FB    283          CMPB    (AP), #FINL_ACS_DESC/4   ; Do we have this argument?
                      67    1F  00FE    284          BLSSU   30$                      ; Nope
                   14 AC    D5  0100    285          TSTL    FINL_ACS_DESC(AP)        ; Yep, but was it really supplied?
                      62    13  0103    286          BEQL    30$                      ; Not really there...
                7E    66    7D  0105    287          MOVQ    (R6), -(SP)              ; It's there, save current desc
                66    54    7D  0108    288 10$:     MOVQ    R4, (R6)                 ; Set new string length/pointer
    04 B6   66    FEF0 CF    02  39  010B    289          MATCHC  #2, DOUBLE_COLON, (R6), @4(R6) ; Find a node name
                54    52    7D  0113    290          MOVQ    R2, R4                   ; Save length/pointer past node name
       63   52    FEE5 CF    02  39  0116    291          MATCHC  #2, DOUBLE_COLON, R2, (R3) ; Another node name?
                      E9    13  011D    292          BEQL    10$                      ; Yes, loop...
                66    54    C2  011F    293          SUBL    R4, (R6)                 ; Isolate the last node name
             04 B6    66    22  3A  0122    294          LOCC    #^A/"/, (R6), @4(R6)     ; Find the ACS if any
                54    50    7D  0127    295          MOVQ    R0, R4                   ; Save descriptor of ACS
                52    50    D0  012A    296          MOVL    R0, R2                   ; Really an ACS?
                      0B    13  012D    297          BEQL    20$                      ; Nope
                      54    D7  012F    298          DECL    R4                       ; Yep, skip the leading quote in count
                      55    D6  0131    299          INCL    R5                       ;  and bump pointer over it
             65       54    22  3A  0133    300          LOCC    #^A/"/, R4, (R5)         ; Find the trailing quote
                      54    50  C2  0137    301          SUBL    R0, R4                   ;  and remove rest of string
                      66    54  7D  013A    302 20$:     MOVQ    R4, (R6)                 ; Load descriptor of ACS
                      66    7F  013D    303          PUSHAQ  (R6)                     ; Arg #2 is the src string desc
                   14 BC    7F  013F    304          PUSHAQ  @FINL_ACS_DESC(AP)       ; Arg #1 is the dst string desc
             FEF6  CF      02  FB  0142    305          CALLS   #2, STRING_COPY          ; Go copy final ACS to caller's desc
                66    8E    7D  0147    306          MOVQ    (SP)+, (R6)              ; Restore original descriptor
                   0175     30  014A    307          BSBW    LOWBIT_ELSE_DIE          ; Check the completion code
                      52    D5  014D    308          TSTL    R2                       ; Really any ACS?
                      16    13  014F    309          BEQL    30$                      ; Nope
                54    02    C0  0151    310          ADDL    #2, R4                   ; Yep, add back in the quotes
                66    54    C2  0154    311          SUBL    R4, (R6)                 ;  and decrease original string's size
                      55    D7  0157    312          DECL    R5                       ; Correct pointer for initial quote
          52    55    04 A6   C3  0159    313          SUBL3   4(R6), R5, R2            ; Find size of left half of string
             52    66    52   C3  015E    314          SUBL3   R2, (R6), R2             ;  then size of new right half
       65    6544    52   28  0162    315          MOVC    R2, (R5)[R4], (R5)       ; Shuffle up string...
                              0167    316 30$:                                     ; Continue
                              0167    317
                              0167    318 CHK_PASS_PROTO:                          ; Check for needing pass through
                   F8 AD      DF  0167    319          PUSHAL  WAIT_EFN(FP)             ; Arg #1 is the EFN return longword
          00000000'GF      01  FB  016A    320          CALLS   #1, G^LIB$GET_EF         ; Go allocate us an EFN
                     014E     30  0171    321          BSBW    LOWBIT_ELSE_DIE          ; Check the completion code
                        7E    D4  0174    322          CLRL    -(SP)                    ; Guess at a direct connection
    04 B6   66    FE85 CF    02  39  0176    323          MATCHC  #2, DOUBLE_COLON, (R6), @4(R6) ; Find first node name
                54    52    7D  017E    324          MOVQ    R2, R4                   ; Save size/pointer to rest of string
       63   52    FE7A CF    02  39  0181    325          MATCHC  #2, DOUBLE_COLON, R2, (R3) ; Is there another node name?
                      4D    12  0188    326          BNEQ    20$                      ; Nothing more, a direct connection
             5E    FF08 CE    9E  018A    327          MOVAB   -<IO_BUF_LEN-4-4>(SP), SP ; Make room for I/O buffer on stack
                      7E    7C  018F    328          CLRQ    -(SP)                    ; Preset number of hops to zero, etc.
                08    6C    91  0191    329          CMPB    (AP), #NUM_HOPS_BYTE/4   ; Do we have this argument?
                      0A    1F  0194    330          BLSSU   10$                      ; Nope
                   20 AC      D5  0196    331          TSTL    NUM_HOPS_BYTE(AP)        ; Yep, but was it really supplied?
                      05    13  0199    332          BEQL    10$                      ; Not really there...
```

```
                                                E 8
                 04 AE    20 BC    90  019B    333          MOVB    @NUM_HOPS_BYTE(AP), 4(SP) ; Yep, set number of hops to far
                   7E    54  01    C1  01A0    334 10$:     ADDL3   #1, R4, -(SP)             ; Save path list's length
                         66  54    C2  01A4    335          SUBL    R4, (R6)                  ; Take path list size from string
09 AE    00FF 8F    00    65  54    2C  01A7    336          MOVC5   R4, (R5), #0, #IO_BUF_LEN-1, 8+1(SP) ; Move list to I/O buffer
                         66  02    C2  01B0    337          SUBL    #2, (R6)                  ; Take the double colon from node name
                   04 B6   66  22    3A  01B3    338          LOCC    #^A/"/, (R6), @4(R6)      ; Find any ACS in node name
                         66  50    C2  01B8    339          SUBL    R0, (R6)                  ;  and take it away
                         66  1C    C0  01BB    340          ADDL    #PASS_OBJECT_LEN, (R6)    ; Update count for object type
               61    FE3F CF    1C  28  01BE    341          MOVC    #PASS_OBJECT_LEN, PASS_OBJECT, (R1) ;  and move it in
                         07  6C    91  01C4    342          CMPB    (AP), #BUF_SIZE_WORD/4    ; Do we have this argument?
                             0E    1F  01C7    343          BLSSU   20$                       ; Nope
                           1C AC    D5  01C9    344          TSTL    BUF_SIZE_WORD(AP)         ; Yep, but was it really supplied?
                             09    13  01CC    345          BEQL    20$                       ; Not really there...
                      EE A3      02  90  01CE    346          MOVB    #2, PASS_OBJECT_CNT(R3)   ; Set connect data count to two bytes
                   EF A3    1C BC    B0  01D2    347          MOVW    @BUF_SIZE_WORD(AP), PASS_OBJECT_CNT+1(R3) ;  and fill them in
                         FE43 CF    9F  01D7    348 20$:     PUSHAB  NET_DEV_NAM               ; Form a descriptor to
                             06  DD    01DB    349          PUSHL   #NET_DEV_NAM_LEN          ;  the DECnet "device name"
                         50  6E  7E    01DD    350          MOVAQ   (SP), R0                  ;  and get a pointer to it
                           10 57  E8    01E0    351          BLBS    R7, 30$                   ; Are we doing a mailbox?
                                        01E3    352          $ASSIGN_S -                       ; Assign a channel
                                        01E3    353                  DEVNAM = (R0), -          ;  use the DECnet "device name"
                                        01E3    354                  CHAN = @NET_CHAN_WORD(AP) ;  place channel number back here
                             7E  7C     01E3                         CLRQ    -(SP)
                           08 BC  3F    01E5                         PUSHAW  @NET_CHAN_WORD(AP)
                             60  7F     01E8                         PUSHAQ  (R0)
                00000000'GF  04  FB     01EA                         CALLS   #4,G^SYS$ASSIGN
                             17  11     01F1    355          BRB     40$                       ; Go check the completion status
                                        01F3    356
                           0C BC  3F    01F3    357 30$:     PUSHAW  @MBX_CHAN_WORD(AP)        ; Arg #5 is the mailbox channel word
                           08 BC  3F    01F6    358          PUSHAW  @NET_CHAN_WORD(AP)        ; Arg #4 is the DECnet channel word
                      FE27 CF     DF    01F9    359          PUSHAL  CREMBX_BUFQUO             ; Arg #3 is the buffer quota
                      FE27 CF     DF    01FD    360          PUSHAL  CREMBX_MAXMSG             ; Arg #2 is the maximum message
                             60   7F    0201    361          PUSHAQ  (R0)                      ; Arg #1 is the DECnet "device name"
             00000000'GF     05   FB    0203    362          CALLS   #5, G^LIB$ASN_WTH_MBX     ; Assign channel with a mailbox
                        36   50   E9    020A    363 40$:     BLBC    R0, 50$                   ; Go die unless success completion
                      54    66   7E    020D    364          MOVAQ   (R6), R4                  ; Get pointer to connect information
                      55    6E   7E    0210    365          MOVAQ   (SP), R5                  ;  and a pointer to an IOSB
                                        0213    366          $QIOW_S -                         ; Request connect initiate
                                        0213    367                  CHAN = @NET_CHAN_WORD(AP), - ;  on the destination
                                        0213    368                  FUNC = S^#IO$_ACCESS, -   ;  using the access function
                                        0213    369                  EFN = WAIT_EFN(FP), -     ;  waiting for it to complete
                                        0213    370                  IOSB = (R5), -            ;  use an IOSB
                                        0213    371                  P2 = R4                   ;  use the connect information
                             7E  7C     0213                         CLRQ    -(SP)
                             7E  7C     0215                         CLRQ    -(SP)
                             54  DD     0217                         PUSHL   R4
                             00  DD     0219                         PUSHL   #0
                             7E  7C     021B                         CLRQ    -(SP)
                             65  7F     021D                         PUSHAQ  (R5)
                      7E     00' 3C     021F                         MOVZWL  S^#IO$_ACCESS,-(SP)
                   7E    08 BC  3C     0222                         MOVZWL  @NET_CHAN_WORD(AP),-(SP)
                         F8 AD  DD     0226                         PUSHL   WAIT_EFN(FP)
             00000000'GF    0C  FB     0229                         CALLS   #12,G^SYS$QIOW
                        10   50   E9    0230    372          BLBC    R0, 50$                   ; Go die unless success completion
                      50    65   3C    0233    373          MOVZWL  (R5), R0                  ; Get the I/O completion code
                      5E    08   C0    0236    374          ADDL    #8, SP                    ;  and pop the IOSB from stack
                      00'   50   D1    0239    375          CMPL    R0, S^#SS$_NORMAL         ; Is it simply a success completion?
```

```
                     05   12  023C    376          BNEQ    50$                             ; Nope
               50    0000'8F   3C  023E    377          MOVZWL  #SS$_REMOTE, R0             ; Yep, change to connection confirmed
                  FC AD    50   D0  0243    378 50$:     MOVL    R0, RET_STATUS(FP)          ; Store assign's completion code
      04 B6    66    FDB4 CF   02   39  0247    379          MATCHC  #2, DOUBLE_COLON, (R6), @4(R6) ; Find node name again
                           52   02   C0  024F    380          ADDL    #2, R2                  ; Put double colon back into remainder
                           66   52   C2  0252    381          SUBL    R2, (R6)                ;  and take away all but node name
                  04 B6    66   22   3A  0255    382          LOCC    #^A/"/, (R6), @4(R6)    ; Scan for an ACS
                           66   50   C2  025A    383          SUBL    R0, (R6)                ;  and take it away
                     61    3A3A 8F   B0  025D    384          MOVW    #^A/::/, (R1)           ; Add a double colon to it
                           66   02   C0  0262    385          ADDL    #2, (R6)                ;  and count the double colon
                        21 FC AD   E8  0265    386          BLBS    RET_STATUS(FP), 60$     ; Did the channel assign make it?
                                       0269    387          $DASSGN_S -                     ; Deassign a channel
                                       0269    388                  CHAN = @NET_CHAN_WORD(AP) ;  from the destination
                        7E   08 BC   3C  0269                        MOVZWL  @NET_CHAN_WORD(AP),-(SP)
               00000000'GF    01   FB  026D                        CALLS   #1,G^SYS$DASSGN
                           08 BC   B4  0274    389          CLRW    @NET_CHAN_WORD(AP)      ; Clear the channel number word
                           15 57   E9  0277    390          BLBC    R7, EXIT                ; Using a mailbox?
                                       027A    391          $DASSGN_S -                     ; Deassign a channel
                                       027A    392                  CHAN = @MBX_CHAN_WORD(AP) ;  from the mailbox
                        7E   0C BC   3C  027A                        MOVZWL  @MBX_CHAN_WORD(AP),-(SP)
               00000000'GF    01   FB  027E                        CALLS   #1,G^SYS$DASSGN
                           0C BC   B4  0285    393          CLRW    @MBX_CHAN_WORD(AP)      ; Clear mailbox channel number word
                              05   11  0288    394          BRB     EXIT                    ; Go exit
                                       028A    395
                     50    6E   D0  028A    396 60$:     MOVL    (SP), R0                ; A path list to send?
                              60   12  028D    397          BNEQ    DO_PASS_PROTO           ; Yep, go do it
                                       028F    398
                                       028F    399 EXIT:                                   ; Exit after clean ups
                     04    6C   91  028F    400          CMPB    (AP), #RSL_PATH_DESC/4  ; Do we have this argument?
                              0F   1F  0292    401          BLSSU   10$                     ; Nope
                        10    AC   D5  0294    402          TSTL    RSL_PATH_DESC(AP)       ; Yep, but was it really supplied?
                              0A   13  0297    403          BEQL    10$                     ; Not really there...
                              66   7F  0299    404          PUSHAQ  (R6)                    ; Arg #2 is the src string desc
                           10 BC   7F  029B    405          PUSHAQ  @RSL_PATH_DESC(AP)      ; Arg #1 is the dst string desc
                  FD9A CF     02   FB  029E    406          CALLS   #2, STRING_COPY         ; Go copy result path to caller's desc
                        F8 AD    D5  02A3    407 10$:     TSTL    WAIT_EFN(FP)            ; Did we allocate us an EFN?
                              0A   13  02A6    408          BEQL    20$                     ; Nope
                        F8 AD    DF  02A8    409          PUSHAL  WAIT_EFN(FP)            ; Arg #1 is the allocated EFN longword
               00000000'GF    01   FB  02AB    410          CALLS   #1, G^LIB$FREE_EF       ; Go free up our EFN
                     50    FC AD    D0  02B2    411 20$:     MOVL    RET_STATUS(FP), R0      ; Get the return status
                              04   02B6    412          RET                             ; Return.
                                       02B7    413
                                       02B7    414 .ENABLE LSB
                                       02B7    415
                                       02B7    416 LOWBIT_ELSE_END:                        ; End unless success status
                        34 50    E8  02B7    417          BLBS    R0, 10$                 ; Exit if success
                        05 54    E9  02BA    418          BLBC    R4, LOWBIT_ELSE_DIE     ; Have we gotten any messages?
                  50    0000'8F   3C  02BD    419          MOVZWL  #SS$_REJECT, R0         ; Yes, change into connect reject
                                       02C2    420 LOWBIT_ELSE_DIE:                        ; Die unless success status
                        29 50    E8  02C2    421          BLBS    R0, 10$                 ; Exit if success
                     FC AD    50   D0  02C5    422          MOVL    R0, RET_STATUS(FP)      ; Else set the status code
                     50    08 BC   3C  02C9    423          MOVZWL  @NET_CHAN_WORD(AP), R0  ; Get the assigned channel number
                              C0   13  02CD    424          BEQL    EXIT                    ; None, just go exit
                                       02CF    425          $DASSGN_S -                     ; Deassign a channel
                                       02CF    426                  CHAN = R0               ;  from the destination
                        7E    50   3C  02CF                        MOVZWL  R0,-(SP)
               00000000'GF    01   FB  02D2                        CALLS   #1,G^SYS$DASSGN
```

```
                B3 57    E9  02D9    427          BLBC    R7, EXIT                        ; Using a mailbox?
          50    0C BC    3C  02DC    428          MOVZWL  @MBX_CHAN_WORD(AP), R0          ; Get the assigned mailbox channel
             AD    13    02E0    429          BEQL    EXIT                            ; None
                         02E2    430          $DASSGN_S -                             ; Deassign a channel
                         02E2    431                  CHAN = R0                       ;  from the mailbox
             7E    50    3C  02E2                     MOVZWL  R0,-(SP)
00000000'GF    01    FB  02E5                     CALLS   #1,G^SYS$DASSGN
             A1    11    02EC    432          BRB     EXIT                            ;   then go exit
                         02EE    433
                   05    02EE    434 10$:     RSB                                     ; Exit
                         02EF    435
                         02EF    436 .DISABLE LSB
                         02EF    437
                         02EF    438 DO_PASS_PROTO:                                   ; Do the pass through protocol
          55    6E    7E  02EF    439          MOVAQ   (SP), R5                        ; Get pointer to the IOSB, buffer
                         02F2    440          $QIOW_S -                               ; Write the path list
                         02F2    441                  CHAN = @NET_CHAN_WORD(AP), - ;  to the destination
                         02F2    442                  FUNC = S^#IO$_WRITEVBLK, - ;  writing obviously
                         02F2    443                  EFN = WAIT_EFN(FP), -   ;  waiting for it to complete
                         02F2    444                  IOSB = (R5), -          ;  use an IOSB
                         02F2    445                  P1 = 8(R5), -           ;  use pre-loaded I/O buffer
                         02F2    446                  P2 = R0                 ;   with this preset size
             7E    7C  02F2                     CLRQ    -(SP)
             7E    7C  02F4                     CLRQ    -(SP)
             50    DD  02F6                     PUSHL   R0
          08 A5    DF  02F8                     PUSHAL  8(R5)
             7E    7C  02FB                     CLRQ    -(SP)
             65    7F  02FD                     PUSHAQ  (R5)
       7E    00'   3C  02FF                     MOVZWL  S^#IO$_WRITEVBLK,-(SP)
       7E    08 BC    3C  0302                 MOVZWL  @NET_CHAN_WORD(AP),-(SP)
          F8 AD    DD  0306                     PUSHL   WAIT_EFN(FP)
00000000'GF    0C    FB  0309                     CALLS   #12,G^SYS$QIOW
             B0    10    0310    447          BSBB    LOWBIT_ELSE_DIE                 ; Go check the completion code
          50    65    3C  0312    448          MOVZWL  (R5), R0                        ; Get the I/O completion code
             AB    10    0315    449          BSBB    LOWBIT_ELSE_DIE                 ; Go check the completion code
             54    D4    0317    450          CLRL    R4                              ; Say no messages received yet
                         0319    451 10$:     $QIOW_S -                               ; Read a response
                         0319    452                  CHAN = @NET_CHAN_WORD(AP), - ;  from the destination
                         0319    453                  FUNC = S^#IO$_READVBLK, - ;  reading obviously
                         0319    454                  EFN = WAIT_EFN(FP), -   ;  waiting for it to complete
                         0319    455                  IOSB = (R5), -          ;  use an IOSB
                         0319    456                  P1 = 8(R5), -           ;  use the I/O buffer
                         0319    457                  P2 = #IO_BUF_LEN        ;   with its length
             7E    7C  0319                     CLRQ    -(SP)
             7E    7C  031B                     CLRQ    -(SP)
       00000100 8F    DD  031D                     PUSHL   #IO_BUF_LEN
          08 A5    DF  0323                     PUSHAL  8(R5)
             7E    7C  0326                     CLRQ    -(SP)
             65    7F  0328                     PUSHAQ  (R5)
       7E    00'   3C  032A                     MOVZWL  S^#IO$_READVBLK,-(SP)
       7E    08 BC    3C  032D                 MOVZWL  @NET_CHAN_WORD(AP),-(SP)
          F8 AD    DD  0331                     PUSHL   WAIT_EFN(FP)
00000000'GF    0C    FB  0334                     CALLS   #12,G^SYS$QIOW
          FF79    30  033B    458          BSBW    LOWBIT_ELSE_END                 ; Go check the completion code
          50    65    3C  033E    459          MOVZWL  (R5), R0                        ; Get the I/O completion code
          FF73    30  0341    460          BSBW    LOWBIT_ELSE_END                 ; Go check the completion code
       50    02 A5    3C  0344    461          MOVZWL  2(R5), R0                       ; Get size of response
```

```
                                                H  8

                CF   13  0348    462          BEQL    10$                     ; No length??
                50   D7  034A    463          DECL    R0                      ; Remove status code from message size
          51    08   A5  034C    464          MOVAB   8(R5), R1               ; Point to message in the I/O buffer
             52     81  034A    465          MOVZBL  (R1)+, R2               ;  and extract its status code
             01     52  0353    466          CMPB    R2, #STS$K_SUCCESS      ; Is it success?
                32   13  0356    467          BEQL    30$                     ; We're done if success
             54     01  D0  0358    468          MOVL    #1, R4                  ; Else say some message(s) received
             06     6C  91  035B    469          CMPB    (AP), #MSG_ROUT_ADDR/4  ; Do we have this argument?
                B9   1F  035E    470          BLSSU   10$                     ; Nope
             18   AC  D5  0360    471          TSTL    MSG_ROUT_ADDR(AP)       ; Yep, but was it really supplied?
                B4   13  0363    472          BEQL    10$                     ; Not really there...
                50   D5  0365    473          TSTL    R0                      ; Anything to look at in the message?
                09   13  0367    474          BEQL    20$                     ; Nope??
             25     61  91  0369    475          CMPB    (R1), #^A/%/            ; Yep, a leading percent ("%")?
                04   12  036C    476          BNEQ    20$                     ; No leading percent, all set
                50   D7  036E    477          DECL    R0                      ; Remove the percent from the count
                51   D6  0370    478          INCL    R1                      ;  and skip the pointer over it
             7E     50  7D  0372    479 20$:     MOVQ    R0, -(SP)               ; Put me sage descriptor onto stack
                6E   7F  0375    480          PUSHAQ  (SP)                    ;  then stack the address of that desc
                01   DD  0377    481          PUSHL   #1                      ; FAO count is one
                52   C9  0379    482          BISL3   R2, -                   ; Use the severity level .OR.'d into
                         037B    483                  #<LIB$_STRTRU&STS$M_FAC_NO>!- ;  a facility code of "LIB"
          7E  00001130'8F        037B    484                       SHR$_TEXT, -(SP) ;  and a general text shared message
           18 BC   03  FB  0381    485          CALLS   #3, @MSG_ROUT_ADDR(AP)  ; Call for message output ($PUTMSG)
              5E   08  C0  0385    486          ADDL    #8, SP                  ; Clean up the stack again
                8F   11  0388    487          BRB     10$                     ;  then loop for more...
                         038A    488
             52     86  7D  038A    489 30$:     MOVQ    (R6)+, R2               ; Get descriptor of connection node
          5E 51     52  C3  038D    490          SUBL3   R2, R1, SP              ; Back up stack for the node name
             76     6E  9E  0391    491          MOVAB   (SP), -(R6)             ; Store new address of resultant path
             76     50  C0  0394    492          ADDL    R0, -(R6)               ;  and path's length
          6E 63     52  28  0397    493          MOVC    R2, (R3), (SP)          ; Move connection node to path list
                FEF1   31  039B    494          BRW     EXIT                    ; Go exit...
                         039E    495
                         039E    496 .END
```

```
$$.TMP1                 =  00000001
$$.TMP2                 =  00000062
$$T1                    =  00000001
BUF_SIZE_WORD           =  0000001C
CHK_PASS_PROTO             00000167 R      02
CREMBX_BOFQUO              00000024 R      02
CREMBX_MAXMSG              00000028 R      02
DEV_NAME_DESC           =  00000004
DOUBLE_COLON               00000000 R      02
DO_PASS_PROTO              000002EF R      02
EXIT                       0000028F R      02
FAB$B_BID               =  00000000
FAB$B_BLN               =  00000001
FAB$B_DNS               =  00000035
FAB$B_FNS               =  00000034
FAB$C_BID               =  00000003
FAB$C_BLN               =  00000050
FAB$L_FNA               =  0000002C
FAB$L_NAM               =  00000028
FAB$L_STV               =  0000000C
FINL_ACS_DESC           =  00000014
GET_FINAL_ACS              000000FB R      02
IO$_ACCESS                 ********   X    02
IO$_READVBLK               ********   X    02
IO$_WRITEVBLK              ********   X    02
IO_BUF_LEN              =  00000100
LIB$ANALYZE_SDESC          ********   X    02
LIB$ASN_WTH_MBX            ********   X    02
LIB$FREE_EF                ********   X    02
LIB$GET_EF                 ********   X    02
LIB$SIG_TO_RET             ********   X    02
LIB$_STRTRU                ********   X    02
LOWBIT_ELSE_DIE            000002C2 R      02
LOWBIT_ELSE_END            000002B7 R      02
MBX_CHAN_WORD           =  0000000C
MSG_ROUT_ADDR           =  00000018
NAM$B_BID               =  00000000
NAM$B_BLN               =  00000001
NAM$B_ESL               =  0000000B
NAM$B_ESS               =  0000000A
NAM$B_NODE              =  00000038
NAM$B_NOP               =  00000008
NAM$C_BID               =  00000002
NAM$C_BLN               =  00000060
NAM$C_MAXRSS            =  000000FF
NAM$L_ESA               =  0000000C
NAM$L_NODE              =  00000040
NAM$M_PWD               =  00000001
NET_CHAN_WORD           =  00000008
NET_DEV_NAM                0000001E R      02
NET_DEV_NAM_LEN         =  00000006
NUM_HOPS_BYTE           =  00000020
PARSE_STRING               00000066 R      02
PASS_OBJECT                00000002 R      02
PASS_OBJECT_CNT         =  FFFFFFEE
PASS_OBJECT_LEN         =  0000001C
RET_STATUS              =  FFFFFFFC
RMS$_CHN                   ********   X    02
RMS$_NOD                   ********   X    02
RSL_PATH_DESC           =  00000010
SHR$_TEXT               =  00001130
SS$_NORMAL                 ********   X    02
SS$_REJECT                 ********   X    02
SS$_REMOTE                 ********   X    02
STR$COPY_DX                ********   X    02
STRING_COPY                0000003D R      02
STRING_FETCH               0000002C R      02
STS$K_SUCCESS           =  00000001
STS$M_FAC_NO            =  0FFF0000
SYS$ASSIGN                 ********  GX    02
SYS$DASSGN                 ********  GX    02
SYS$PARSE                  ********  GX    02
SYS$QIOW                   ********  GX    02
UNS$NET_CONNECT            0000004E RG     02
WAIT_EFN                =  FFFFFFF8
```

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
_LIB$CODE	0000039E (926.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.07	00:00:00.25
Command processing	134	00:00:00.51	00:00:03.11
Pass 1	231	00:00:06.59	00:00:14.70
Symbol table sort	0	00:00:00.61	00:00:01.43
Pass 2	130	00:00:01.71	00:00:04.39
Symbol table output	12	00:00:00.09	00:00:00.08
Psect synopsis output	5	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	551	00:00:09.60	00:00:23.99

The working set limit was 900 pages.
40097 bytes (79 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 528 non-local and 18 local symbols.
496 source lines were read in Pass 1, producing 15 object records in Pass 2.
22 pages of virtual memory were used to define 20 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYSLIB]STARLET.MLB;2	17

677 GETS were required to define 17 macros.

There were no errors, warnings or information messages.

MACRO/DISA=TRACE/LIS=LIS$:LIBNETCON/OBJ=OBJ$:LIBNETCON MSRC$:LIBNETCON/UPDATE=(ENH$:LIBNETCON)

READOBJ
LIS

SCREEN
LIS

LIBEXECLI
LIS

MOUNTMSG
LIS

PASMSG
LIS

LIBFILPRO
LIS

LIBNETCON
LIS

LIBEXTCON
LIS

LIBMERGE
LIS

LNKMSG
LIS

REPORTIOE
LIS

LIBUNLFIL
LIS

LIBFIDNAM
LIS

MATCHNAME
LIS

OPCMSG
LIS

PLIMSG
LIS