B 9

```
NNN         NNN  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT     AAAAAAAAA        CCCCCCCCCCCC  PPPPPPPPPPPP
NNN         NNN  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT     AAAAAAAAA        CCCCCCCCCCCC  PPPPPPPPPPPP
NNN         NNN  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT     AAAAAAAAA        CCCCCCCCCCCC  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  EEEEEEEEEEEE           TTT        AAA         AAA  CCC              PPPPPPPPPPPP
NNN   NNN   NNN  EEEEEEEEEEEE           TTT        AAA         AAA  CCC              PPPPPPPPPPPP
NNN   NNN   NNN  EEEEEEEEEEEE           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     CCCCCCCCCCCC  PPP
NNN         NNN  EEEEEEEEEEEEEEE        TTT        AAA         AAA     CCCCCCCCCCCC  PPP
NNN         NNN  EEEEEEEEEEEEEEE        TTT        AAA         AAA     CCCCCCCCCCCC  PPP
```

```
_S
Ps
NE
NE
NE
NE
$R
```

FILEID**NETPROCRE

K 13

```
NN      NN  EEEEEEEEEE  TTTTTTTTTT  PPPPPPPP    RRRRRRRR      OOOOOO      CCCCCCCC  RRRRRRRR    EEEEEEEEEE
NN      NN  EEEEEEEEEE  TTTTTTTTTT  PPPPPPPP    RRRRRRRR      OOOOOO      CCCCCCCC  RRRRRRRR    EEEEEEEEEE
NN      NN  EE              TT      PP      PP  RR      RR  OO      OO  CC          RR      RR  EE
NN      NN  EE              TT      PP      PP  RR      RR  OO      OO  CC          RR      RR  EE
NNNN    NN  EE              TT      PP      PP  RR      RR  OO      OO  CC          RR      RR  EE
NNNN    NN  EE              TT      PP      PP  RR      RR  OO      OO  CC          RR      RR  EE
NN  NN  NN  EEEEEEEE        TT      PPPPPPPP    RRRRRRRR    OO      OO  CC          RRRRRRRR    EEEEEEEE
NN  NN  NN  EEEEEEEE        TT      PPPPPPPP    RRRRRRRR    OO      OO  CC          RRRRRRRR    EEEEEEEE
NN    NNNN  EE              TT      PP          RR  RR      OO      OO  CC          RR  RR      EE
NN    NNNN  EE              TT      PP          RR  RR      OO      OO  CC          RR  RR      EE
NN      NN  EE              TT      PP          RR    RR    OO      OO  CC          RR    RR    EE              ....
NN      NN  EE              TT      PP          RR    RR    OO      OO  CC          RR    RR    EE              ....
NN      NN  EEEEEEEEEE      TT      PP          RR      RR    OOOOOO      CCCCCCCC  RR      RR  EEEEEEEEEE      ....
NN      NN  EEEEEEEEEE      TT      PP          RR      RR    OOOOOO      CCCCCCCC  RR      RR  EEEEEEEEEE      ....

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            .TITLE  NETPROCRE - Process creation
0000            .IDENT  'V04-000'
0000            .DEFAULT DISPLACEMENT,WORD
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000
0000
0000    ;++
0000    ; FACILITY:     NETWORK ACP
0000    ;
0000    ; ABSTRACT:
0000    ;       THIS MODULE PERFORMS PROCESS CREATION FOR AN INBOUND CONNECT.
0000    ;
0000    ; ENVIRONMENT:
0000    ;       MODE = KERNEL
0000    ;
0000    ; AUTHOR:       SCOTT G. DAVIS, CREATION DATE: 10-AUG-77
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-024 ADE0039         Alan D. Eldridge        18-Jul-1984
0000    ;               When looking for a free XWB slot, don't allow either byte of
0000    ;               the local link number to be equal the character '"' since
0000    ;               that results in some non-intelligent NCB parsers to break.
0000    ;
0000    ;       V03-025 PRB0340         Paul Beck       18-Jul-1984  16:10
0000    ;               Test against LGI$_INVPWD for invalid access instead of magic number.
0000    ;
0000    ;       V03-024 ADE0038         Alan D. Eldridge        25-Jun-1984
0000    ;               Change SS$_NOLINKS to SS$_CONNECFAIL on problems finding
0000    ;               or creating logical-link resources.
0000    ;
0000    ;       V03-023 RNG0023         Rod Gamache             12-Jun-1984
0000    ;               Change calling conventions for calls to NODE COUNTER
0000    ;               BLOCK access routines.
```

```
0000    ;
0000    ;       V03-022 PRB0331         Paul Beck       1-May-1984  20:19
0000    ;               1.      Look for EPID instead of IPID in OBI$L_PID
0000    ;               2.      Fix callers of NET$DELIVER_CI to set up R0 correctly.
0000    ;
0000    ;       V03-021 ADE0001         Alan D. Eldridge        11-Apr-1984
0000    ;               When comparing remote link addresses in NET$PROC_XWB, ignore
0000    ;               an address of zero.
0000    ;
0000    ;       V03-019 PRB0317         Paul Beck       8-Mar-1984  17:36
0000    ;               Force created network processes to use DCL as their default
0000    ;               CLI, independent of the default CLI for the specified account.
0000    ;               Fix bug in ADE0035.
0000    ;
0000    ;       V018    ADE0035         Alan D. Eldridge        14-Feb-1984
0000    ;               Create LLI entry when receive notification of a new XWB.
0000    ;
0000    ;       V017    RNG0017         Rod Gamache     7-Feb-1984
0000    ;               Fix initialization of local storage in NET$DELIVER_CI routine.
0000    ;
0000    ;       V016    TMH0016         Tim Halvorsen   23-Jun-1983
0000    ;               Fix selection of waiting network processes so
0000    ;               that processes which were activated with different
0000    ;               default accounts (using default accounts on different
0000    ;               objects) are correctly selected.
0000    ;
0000    ;       V015    RNG0015         Rod Gamache     20-Apr-1983
0000    ;               Fix branch destinations out of range.
0000    ;
0000    ;       V014    TMH0014         Tim Halvorsen   03-Mar-1983
0000    ;               If requested object name starts with a "$", then use
0000    ;               a default filespec of SYS$SYSTEM (rather than SYS$LOGIN)
0000    ;               since objects with a "$" are reserved to DEC.
0000    ;               Allow STARTUP_OBJ to be called with an object name
0000    ;               as well as a number.
0000    ;               Notify new DLE module of process termination.
0000    ;
0000    ;       V013    TMH0013         Tim Halvorsen   14-Feb-1983
0000    ;               Remove node proxy access parameter.
0000    ;               Add support for EPIDs.
0000    ;               Return IPID of SPI database key in IOSB of DECLSERV QIO
0000    ;               to NETSERVER process.
0000    ;
0000    ;       V012    RNG0012         Rod Gamache     26-Jan-1983
0000    ;               Fix bug in NET$DELIVER_CI which doesn't check status
0000    ;               for success on call to memory allocation routine.
0000    ;
0000    ;       V011    TMH0011         Tim Halvorsen   28-Dec-1982
0000    ;               Add routine to break all links for a given process.
0000    ;               Do not store NCB, SFI or PNM into SPI until the link
0000    ;               is actually given to the process, and not when the
0000    ;               process is created.
0000    ;
0000    ;       V010    TMH0010         Tim Halvorsen   11-Nov-1982
0000    ;               Fix bug in NETSERVER startup, so that initial connects
0000    ;               which have been tagged for a certain process do not get
0000    ;               inadvertantly given to the another free server process
```

```
0000   115 ;                 for which the logical link was not intended.
0000   116 ;
0000   117 ;        V009     TMH0009         Tim Halvorsen   09-Jul-1982
0000   118 ;                 Make it possible for the network channel to be cleaned
0000   119 ;                 up without any errors.
0000   120 ;                 Add code to report mailbox messages of MSG$_RESET
0000   121 ;                 to the Transport module, so that it can respond to
0000   122 ;                 X.25 circuit resets during datalink startup.
0000   123 ;
0000   124 ;        V008     TMH0008         Tim Halvorsen   16-Jun-1982
0000   125 ;                 Add an entry to the SPI database when creating a network
0000   126 ;                 job, and remove it when we get the termination message.
0000   127 ;                 Add code to transfer connect requests to waiting server
0000   128 ;                 processes, in order to optimize server process creation.
0000   129 ;                 Fix code in process termination to ignore the INHIB_MSG
0000   130 ;                 bit in the final termination status, when making the
0000   131 ;                 determination of whether the object procedure exists or not.
0000   132 ;                 Do not cause a proxy login if the connect format type is
0000   133 ;                 not a 2.  This prevents an 8 byte PID from being sent to
0000   134 ;                 LOGIN for proxy logins.
0000   135 ;
0000   136 ;        V007     TMH0007         Tim Halvorsen   12-Apr-1982
0000   137 ;                 Get address of utility buffer from cell, rather than
0000   138 ;                 referencing a statically defined location.
0000   139 ;                 Modify ACP mailbox dispatching to handle X.25 mailbox
0000   140 ;                 messages, and dispatch them.
0000   141 ;                 Fix a bug in mailbox dispatching, so that if the mailbox
0000   142 ;                 read is canceled or aborted, then the QIO is re-issued.
0000   143 ;                 Make default addressing word relative and remove explicit
0000   144 ;                 addressing specifiers.
0000   145 ;
0000   146 ;        V03-06   ADE0035         A.Eldridge      11-Feb-1982
0000   147 ;                 Move check for specific OBI proxy access state to allow objects
0000   148 ;                 not in the database and with an object number zero to use the
0000   149 ;                 proxy access specified for the TASK OBI.
0000   150 ;
0000   151 ;        V03-05   ADE0034         A.Eldridge      10-Feb-1982
0000   152 ;                 Return error (instead of bug_check) if call to $CREMBX fails.
0000   153 ;
0000   154 ;        V03-04   ADE0033         A.Eldridge      18-Jan-1981
0000   155 ;                 Fix bug in proxy login.  If the access control string received
0000   156 ;                 in the connect message is non-null then don't allow proxy
0000   157 ;                 login.
0000   158 ;
0000   159 ;        V03-03   ADE0032         A.Eldridge      26-Dec-1981
0000   160 ;                 Allow maximum task name of 12 characters in NCB.
0000   161 ;
0000   162 ;        V03-02   ADE0031         A.Eldridge      18-Dec-1981
0000   163 ;                 Make sure that the NCB, the taskname, the process name, and
0000   164 ;                 the access control strings passed to LOGINOUT, are properly
0000   165 ;                 uppercased.
0000   166 ;
0000   167 ;        V03-01   ADE0030         A.Eldridge      30-Nov-1981
0000   168 ;                 Added proxy login (access) support.
0000   169 ;
0000   170 ;        V03-00   ADE0029         A.Eldridge      01-Nov-1981
0000   171 ;                 General cleanup.
```

```
                    0000    172 ;
                    0000    173 ;       V02-17  TMH0017         Tim Halvorsen   04-Sep-1980
                    0000    174 ;               Accept SYS$NET parameter as input to NET$STARTUP_OBJ.
                    0000    175 ;
                    0000    176 ;       V2                      A.Eldridge      01-Jan-1980
                    0000    177 ;               Rewritten for Phase III
                    0000    178 ;---
```

```
                         0000  .SBTTL  DECLARATIONS
                         0000  ;
                         0000  ; INCLUDE FILES:
                         0000  ;
                         0000          $ACCDEF
                         0000          $AQBDEF
                         0000          $STSDEF
                         0000          $DRDEF
                         0000          $MSGDEF
                         0000          $PRVDEF
                         0000          $UCBDEF
                         0000          $IRPDEF
                         0000          $RCBDEF
                         0000          $LLIDEF
                         0000          $LTBDEF
                         0000
                         0000          $CNFDEF
                         0000          $CNRDEF
                         0000          $NFBDEF
                         0000          $NMADEF
                         0000          $XWBDEF
                         0000
                         0000          $WQEDEF
                         0000
                         0000          $NETSYMDEF
                         0000          $NETUPDDEF
                         0000  ;
                         0000  ; MACROS:
                         0000  ;
                         0000
                         0000  ;
                         0000  ; EQUATED SYMBOLS:
                         0000  ;
                         0000
               00000024  0000          CNF = CNF$C_LENGTH                 ; Short name for readability
                         0000
               00000008  0000  STS_M_NOACNT    =    1@3                   ; Do not generate accounting records
               00000040  0000  STS_M_NOAUTH    =    1@6                   ; use caller's privs/quotas at login
               00000080  0000  STS_M_NETLOG    =    1@7                   ; bit no. for network login
               00000096  0000  MBX_MSG_LTH =        150                   ; size of a mailbox message
               0000000C  0000  MAX_TASKNAM =         12                   ; Max size of taskname -- the name
                         0000                                             ; following teh "=" in the NCB
                         0000
                         0000          ASSUME  MBX_MSG_LTH GE ACC$K_TERMLEN
```

```
                   0000  ;
                   0000  ; OWN STORAGE:
                   0000  ;
               00000000             .PSECT  NET_IMPURE,WRT,NOEXE,LONG
                   0000
                   0000             .ALIGN  LONG
                   0000  NET_L_R0:                               ;
          00000000 0000  NET_L_FCT:       .LONG   0              ; Function to pass to NETDRIVER
                   0004  NET_L_R1:                               ;
                   0004  NET_L_LPD:                              ; LPD of line which is starting
          00000000 0004  NET_L_PID:       .LONG   0              ; PID to pass to NETDRIVER
                   0008  NET_L_R2:                               ;
          00000000 0008  NET_L_REASON:    .LONG   0              ; Disconnect reason
                   000C  NET_L_R3:                               ;
          00000000 000C  NET_L_LNK:       .LONG   0              ; Link number
                   0010  NET_L_R4:                               ;
          00000014 0010  NET_A_NCB:       .BLKA   1              ; For saving address of NCB buffer
                   0014  NET_L_R5:                               ;
          00000000 0014  NET_L_UCB:       .LONG   0              ; UCB address to pass to NETDRIVER
                   0018
          00000000 0018  PTR_NCB_BUF:     .LONG   0              ; Address of NCB buffer
          00000000 001C  PTR_CON_BUF:     .LONG   0              ; Address of DELIVER_CI scratch buffer
                   0020
          00000000 0020  NET_A_LLI:       .LONG   0              ; Address of create LLI
 00000000 00000000 0024  NET_Q_NCB:       .QUAD   0              ; NCB descriptor
 00000000 00000000 002C  NET_Q_PRC:       .QUAD   0              ; Process descriptor
 00000000 00000000 0034  NET_Q_TSK:       .QUAD   0              ; Name of file to run
 00000000 00000000 003C  NET_Q_ACC:       .QUAD   0              ; Descriptor for 3 account
                   0044                                          ; strings preceded by flags word
                   0044
          00000005 0044  DET_C_ACC = 5                           ; Buffer for access control strings
              0000 0044  DET_AB_ACC:      .WORD   0              ; for creating detached, privileged
                00 0046                   .BYTE   0              ; processes.  It consists of a flags
                00 0047                   .BYTE   0              ; word followed by 3 null counted
                00 0048                   .BYTE   0              ; strings.
                   0049
                00 0049  OBI_B_PRX:       .BYTE   0              ; OBI proxy access state
                00 004A  INT_B_PRX:       .BYTE   0              ; Internal proxy access state.  This is
                   004B                                          ; set to "none" if any conditions are
                   004B                                          ; detected internally (other than the
                   004B                                          ; values stored in the OBI or NDI)
                   004B                                          ; which would disallow proxy access
                   004B  ;
                   004B  ;  Fields used for termination mailbox creation, message buffering.  Be
                   004B  ;! careful when modifying since some code assumes data ordering without
                   004B  ;! using assumes.
                   004B  ;
                   004B                   .ALIGN  LONG
          0000004E 004C  MBX_CHAN:        .BLKW   1              ; Channel number of mailbox
          00000050 004E  MBX_RDCNT:       .BLKW   1              ; Number of reads posted to mailbox
                   0050  MBX_IOSB:                               ; I/O status block
          00000052 0050                   .BLKW   1              ;  -- status of i/o completion
          00000054 0052  MBX_LEN:         .BLKW   1              ;  -- length of operation here
          00000058 0054  MBX_PID:         .BLKL   1              ;  -- pid of process deleted
                   0058  EXIT_MSG:                               ; Buffer for mailbox message
          0000005A 0058  EXIT_ID:         .BLKW   1              ;  -- message identification
          0000005C 005A                   .BLKW   1              ;  -- not used
```

```
                                    005C   282 NCB_DATA:                                    ; On connect initiates
                           00000060 005C   283 EXIT_CODE:         .BLKL    1                ;  status of message
                           000000F6 0060   284                    .BLKB    MBX_MSG_LTH      ; Leave room for message
                                    00F6   285
                                    00F6   286 NET$GQ_WQE_MBX::                             ; MBX read element
                          000000F6' 00F6   287                    .LONG    .                ; FLINK
                          000000F6' 00FA   288                    .LONG    .-4              ; BLINK
                              0018' 00FE   289                    .WORD    WQE_MBX_LTH      ; Length of entry
                                00' 0100   290                    .BYTE    NET$C_DYN_WQE    ; Structure type
                                 05 0101   291                    .BYTE    WQE$C_SUB_MBX    ; Sub-type is "MBX"
                          000000E6' 0102   292                    .ADDRESS MBX_ACTION       ; Action routine address
                           00000000 0106   293                    .LONG    0                ; AST parameter
                           00000000 010A   294                    .LONG    0                ; "In-use" flag
                           00000018 010E   295 WQE_MBX_LTH = .-NET$GQ_WQE_MBX
                                    010E   296 ;
                                    010E   297 ;  Buffer to get mailbox unit number for $CREPRC argument
                                    010E   298 ;
                           0000011A 010E   299 BBUF:              .BLKL    3                ; Device characteristics
                           0000011C 011A   300 MBX_UNIT:          .BLKW    1                ; Unit number for CREPRC argument
                                    011C   301 ENDBUF:                                      ; Truncate the rest !
                           00000130 011C   302 ZNABUF:            .BLKB    MAX_TASKNAM+8    ; Buffer for building ZNA
                                    0130   303                                             ; the 8 includes 1 byte for the object
                                    0130   304                                             ; number and 7 bytes of slop
                                    0130   305
                                    0130   306
                                    0130   307
                           00000000        308            .PSECT  NET_PURE,NOWRT,NOEXE,LONG
                                    0000   309
                                    0000   310
        54 45 4E 00000008'010E0000' 0000   311 NET_Q_NETPREFIX:.ASCID  "NET"               ; Prefix for unnamed tasks
                           00000005 000B   312 NET_Q_TASKZNA:  .LONG    5                  ; Length of TASK ZNA string
                          00000013' 000F   313                 .ADDRESS TASKZNA            ; Its pointer
                                 00 0013   314 TASKZNA:        .BYTE    0                  ; Object type
                        4B 53 41 54 0014   315                 .ASCII   "TASK"             ; Object name
                                    0018   316 EXIT_BUF:                                   ; Descriptor for channel info
                           0000000E 0018   317                 .LONG    ENDBUF-BBUF        ; Length of buffer
                          0000010E' 001C   318                 .LONG    BBUF               ; Address of buffer
                                    0020   319
                                    0020   320 NET$GQ_MBX_NAME::
50 43 41 54 45 4E 00000028'010E0000' 0020   321                    .ASCID   "NETACP$MBX"     ; Logical name of mailbox
                        58 42 4D 24 002E
                                    0032   322 NET_Q_SYSTEM:
59 53 24 53 59 53 0000003A'010E0000' 0032   323                    .ASCID   "SYS$SYSTEM:"    ; Prefix for reserved objects
                     3A 4D 45 54 53 0040
                                    0045   324 NET_Q_IMAGE:
59 53 24 53 59 53 0000004D'010E0000' 0045   325                    .ASCID   "SYS$SYSTEM:DCL"  ; Login image
               4C 43 44 3A 4D 45 54 53 0053
                                    005B   326 NET_Q_PROC:
59 53 24 53 59 53 00000063'010E0000' 005B   327                    .ASCID   "SYS$SYSTEM:NETSERVER"  ; Network server procedure
56 52 45 53 54 45 4E 3A 4D 45 54 53 0069
                              52 45 0075
                                    0077   328 X25_DEV_NAME:
        41 57 4E 0000007F'010E0000' 0077   329                    .ASCID   "NWA"            ; X.25 device name
                                    0082   330
                           00000000        331            .PSECT  NET_CODE,NOWRT,LONG
                          0000  .SBTTL  NET$PROC_XWB - Process returned XWB
                          0000  ;+
                          0000  ;
                          0000  ;  NETDRIVER has passed us an XWB either to be linked into the LTB and assigned
                          0000  ;  a local logical-link address (upon receiving an incoming connect) or to be
                          0000  ;  unhooked from the LTB and deallocated.
                          0000  ;
                          0000  ;  If both the XWB$W_REMLNK and XWB$W_LOCLNK fields are zero, then this request
                          0000  ;  comes from the NETACP code which handles the IO$_ACCESS request for Connect
                          0000  ;  initiates.
                          0000  ;
                          0000  ;  NETACP is responsible for the LTB maintenance and the XWB linkage in order to
                          0000  ;  avoid any race conditions it may have with NETDRIVER while scanning this list
                          0000  ;
                          0000  ;
                          0000  ;  INPUTS:       R3      XWB pointer
                          0000  ;
                          0000  ;  OUTPUTS:      All registers are clobbered
                          0000  ;
                          0000  ;-
                          0000          .SAVE PSECT
                      00000000          .PSECT  NET_LOCK_CODE,NOWRT,GBL ; Can't tolerate page faults
                          0000          .ENABL  LSB
                          0000
                          0000  NET$PROC_XWB::                          ; Process (deallocate) XWB
      5B    0000'CF   D0  0000          MOVL    NET$GL_CNR_LLI,R11      ; Pick up LLI CNR
               5A     D4  0005          CLRL    R10                     ; No LLI CNF yet
        52   30 A3    D0  0007          MOVL    XWB$L_VCB(R3),R2        ; Get RCB
        55   24 A2    D0  000B          MOVL    RCB$L_PTR_LTB(R2),R5    ; Get LTB
             3E A3    B5  000F          TSTW    XWB$W_LOCLNK(R3)        ; Test local link number
               03     12  0012          BNEQ    2$                      ; If NEQ, XWB being returned
             0087     31  0014          BRW     NEW_LINK                ; If EQL, this is an incoming connect
                          0017  2$:     ;
                          0017          ;
                          0017          ;     Locate and Delete the LLI CNF.  Release hold on counter block
                          0017          ;
                          0017          ;
           58   53    D0  0017          MOVL    R3,R8                   ; Setup XWB address for search
                          001A          $SEARCH eql,lli,l,xwb           ; Find the corresponding LLI, if any
           2E   50    E9  0027          BLBC    R0,10$                  ; If LBC, not found
                          002A                                          ;
               3C     BB  002A          PUSHR   #^M<R2,R3,R4,R5>        ; Save regs
  5E    00000064 8F   C2  002C          SUBL    #100,SP                 ;&Create dummy non-pageable buffer
           56   5E    D0  0033          MOVL    SP,R6                   ; Point to dummy buffer
           54   01    D0  0036          MOVL    #1,R4                   ; Say "zero XWB counters"
           55   53    D0  0039          MOVL    R3,R5                   ; XWB ptr for subr call
            FFC1'     30  003C          BSBW    NET$FLUSH_LLI_CNT       ; Flush LLI and XWB counters to node
                          003F                                          ; counter block
  5E    00000064 8F   C0  003F          ADDL    #100,SP                 ;&Release stack space
               3C     BA  0046          POPR    #^M<R2,R3,R4,R5>        ; Restore regs
                          0048                                          ;
               58     D4  0048          CLRL    R8                      ; Nullify pointer
            FFB3'     30  004A          BSBW    CNF$PUT_FIELD           ; Erase the XWB pointer
            FFB0'     30  004D          BSBW    CNF$DELETE              ; Mark the entry for deletion
            FFAD'     30  0050          BSBW    CNF$PURGE               ; Purge the entry from the database
               5A     D4  0053          CLRL    R10                     ; Forget about the LLI, its gone
            FFA8'     30  0055          BSBW    NET$RELEASE_NDCOU       ; Release hold on counter block
```

```
                             0058  10$:    ;
                             0058          ;
                             0058          ;       This is an old XWB coming back to be removed and deallocated
                             0058          ;
                             0058          ;
       50    3E A3     3C    0058          MOVZWL  XWB$W_LOCLNK(R3),R0          ; Get link number
       50  FC00 8F     AA    005C          BICW    #^C<NET$C_MAXLNK>,R0         ; Clear all but 'index' bits
       50   10 A540    DE    0061          MOVAL   LTB$L_SLOTS(R5)[R0],R0       ; Get link slot
       53       60     D1    0066          CMPL    (R0),R3                      ; Does address match ?
                2F     12    0069          BNEQ    200$                         ; If NEQ, bug
                             006B          DSBINT  #NET$C_IPL                   ; Synchronize with NETDRIVER
                             0071                                               ;
       80       01     B0    0071          MOVW    #1,(R0)+                     ; Set 'available' flag
       60    3E A3     B0    0074          MOVW    XWB$W_LOCLNK(R3),(R0)        ; Store last used link address
       51    E0 A5     9E    0078          MOVAB   -XWB$L_LINK -
                             007C                  +LTB$L_XWB(R5),R1            ; Init for scan
       50       51     D0    007C  20$:    MOVL    R1,R0                        ; Save a copy
       51    2C A1     D0    007F          MOVL    XWB$L_LINK(R1),R1            ; Travel list
       53       51     D1    0083          CMPL    R1,R3                        ; Is this it ?
                F4     12    0086          BNEQ    20$                          ; If not, branch
             2C A3     D0    0088          MOVL    XWB$L_LINK(R3),-             ; Remove it from list
             2C A0           008B                  XWB$L_LINK(R0)               ;
                             008D                                               ;
                             008D          ENBINT                               ; Restore IPL
                             0090                                               ;
                             0090  DEAL_XWB:                                    ; Deallocate XWB
             FF6D'     30    0090          BSBW    NET$DECR_MCOUNT              ; Account for link now gone
       50       53     D0    0093          MOVL    R3,R0                        ; Get block address for call
             FF67'     30    0096          BSBW    NET$DEALLOCATE               ; Deallocate the block
                       05    0099          RSB
                             009A
                             009A  200$:   BUG_CHECK NETNOSTATE,FATAL           ; Else, bad slot address
                             009E
                             009E                  .DSABL  LSB
                             009E
                             009E
                             009E  NEW_LINK:                                    ; Insert new XWB into LTB
                             009E          ;
                             009E          ;
                             009E          ;      Find a free slot in the link table (LTB).  Start from where we left
                             009E          ;      off last time in order to avoid using the same slots over and over
                             009E          ;      again.  This technique increases the interval between re-use of a
                             009E          ;      logical-link number -- i.e., sequence number, slot number.
                             009E          ;
                             009E          ;      Don't allow either byte of the local link number to equal '"' since
                             009E          ;      some non-intelligent NCB parsers mistake that for the end of the
                             009E          ;      NCB.
                             009E          ;
                             009E          ;      The slot vector terminates with a -1 (longword) followed by a
                             009E          ;      0 (longword).
                             009E          ;
                             009E          ;
       50   0000'8F    3C    009E          MOVZWL #SS$_CONNECFAIL,R0            ; Assume failure
       54       65     D0    00A3          MOVL    LTB$L_SLT_NXT(R5),R4         ; Get first slot candidate ptr
          FD    84     E9    00A6  5$:     BLBC    (R4)+,5$                     ; LBC means unavailable
FFFFFFFF 8F     74     D1    00A9          CMPL    -(R4),#-1                    ; Backup and test for end of
                22     12    00B0          BNEQ    10$                          ; NEQ means slot found
```

```
         54    10 A5    DE  00B2              MOVAL   LTB$L_SLOTS(R5),R4           ; Start from top of vector
               FD 84    E9  00B6      7$:     BLBC    (R4)+,7$                     ; LBC means unavailable
  FFFFFFFF 8F     74    D1  00B9              CMPL    -(R4),#-1                    ; Backup and test for end of
                  66    13  00C0              BEQL    200$                         ; EQL means slot not found
      02 A4     0400 8F  A0  00C2      8$:     ADDW    #NET$C_MAXLNK+1,2(R4)        ; Update local link seq #
          22    02 A4   91  00C8              CMPB    2(R4),#^A'"'                 ; Is low byte a double quote?
                  E8    13  00CC              BEQL    7$                           ; If EQL yes, keep scanning
          22    03 A4   91  00CE              CMPB    3(R4),#^A'"'                 ; Is high byte a double quote ?
                  EE    13  00D2              BEQL    8$                           ; If EQL, bump the seq # and try again
                            00D4      10$:    ;
                            00D4              ;
                            00D4              ;       Find position in XWB list.  At the same time, see if this is a
                            00D4              ;       duplicate by matching the remote node address (the remote link
                            00D4              ;       number has not been assigned yet if this is a Connect Initiate).
                            00D4              ;       If its a duplicate, simply deallocate the XWB.
                            00D4              ;
                            00D4              ;
         50    E0 A5    9E  00D4              MOVAB   -XWB$L_LINK -                ;
                            00D8                      +LTB$L_XWB(R5),R0            ; Init for scan
             51    50   D0  00D8      30$:    MOVL    R0,R1                        ; Remember last entry
         50    2C A0    D0  00DB              MOVL    XWB$L_LINK(R0),R0            ; Go to next entry
                  15    13  00DF              BEQL    50$                          ; If EQL, at end of list
               3A A3    B1  00E1              CMPW    XWB$W_REMNOD(R3),-           ; Are we going too far ?
               3A A0        00E4                      XWB$W_REMNOD(R0)             ;
                  0E    1A  00E6              BGTRU   50$                          ; If GTRU yes, stop here
               3C A3    B1  00E8              CMPW    XWB$W_REMLNK(R3),-           ; Is this it ?
               3C A0        00EB                      XWB$W_REMLNK(R0)             ;
                  E9    12  00ED              BNEQ    30$                          ; If NEQ no, continue searching
               3C A3    B5  00EF              TSTW    XWB$W_REMLNK(R3)             ; But, if =0 then no address has been
                  E4    13  00F2              BEQL    30$                          ; assigned; comparison was invalid
                  9A    11  00F4              BRB     DEAL_XWB                     ; ...else duplicate connect
                            00F6      50$:    ;
                            00F6              ;
                            00F6              ;       LTB slot and place in XWB list have been found.  Link XWB into the
                            00F6              ;       LTB and setup local link number.
                            00F6              ;
                            00F6              ;
                            00F6              DSBINT #NET$C_IPL                    ; Synch with NETDRIVER
                            00FC                                                   ;
                  04    AA  00FC              BICW    #XWB$M_STS_SOL,-             ;
               0E A3        00FE                      XWB$W_STS(R3)                ; No longer queued
     3E A3     02 A4    B0  0100              MOVW    2(R4),XWB$W_LOCLNK(R3)       ; Setup local link number
             84    53   D0  0105              MOVL    R3,(R4)+                     ; Store XWB ptr in this slot
             65    54   D0  0108              MOVL    R4,LTB$L_SLT_NXT(R5)         ; Store scan's next starting pt.
          2C A3    50   D0  010B              MOVL    R0,XWB$L_LINK(R3)            ; Link tail of list to current XWB
          2C A1    53   D0  010F              MOVL    R3,XWB$L_LINK(R1)            ; Link XWB to head of list
                            0113
                            0113              ENBINT                               ; Restore IPL
                            0116
               FEE7'    30  0116              BSBW    CREATE_LLI                   ; Create LLI and insert it into database
                            0119                                                   ; Use status as input to NET$DELIVER_CI
               3C A3    B5  0119              TSTW    XWB$W_REMLNK(R3)             ; Connect Initiate ?
                  09    13  011C              BEQL    100$                         ; If EQL yes, return R0 to caller
             56    53   D0  011E              MOVL    R3,R6                        ; Else, copy XWB address
               02B8'    30  0121              BSBW    NET$DELIVER_CI               ; Create LLI, and deliver connect
                            0124                                                   ; notification to some server
         50    01       D0  0124              MOVL    #1,R0                        ; Say "success"
```

```
                         05  0127      100$:   RSB                                    ; Done
                             0128
                             0128      200$:   BUG_CHECK  NETNOSTATE,FATAL
                             012C
                       00000000                .RESTORE_PSECT
                             0000
                             0000
                             0000      CREATE_LLI:                                    ; Create LLI and insert it into the list
                             0000              ;
                             0000              ;  This subroutine in required so that the  "utility buffer"  acquired
                             0000              ;  by the NET$GETUTLBUF co-routine will be released in a timely manner.
                             0000              ;
                             0000              ; NOTE - the NET$ACQUIRE_NDCOU routine needs the utility buffer, so
                             0000              ;  we must not allocate the utility buffer until after we acquire the
                             0000              ;  NDC counter block.
                             0000              ;
              FFFD'  30      0000              BSBW    NET$ACQUIRE_NDCOU              ; Inc. reference level of counter block
              40 50  E9      0003              BLBC    R0,90$                         ; If LBC, problem encountered
              FFF7'  30      0006              BSBW    NET$GETUTLBUF                  ; Get permission to use utility buffer
                             0009                                                     ; - the above is a co-routine call
       5B   0000'CF  D0      0009              MOVL    NET$GL_CNR_LLI,R11             ; Pick up CNR
              FFEF'  30      000E              BSBW    CNF$INIT_UTL                   ; Init "utility buffer" as a CNF
         58      53  D0      0011              MOVL    R3,R8                          ; Get XWB
                             0014              $PUTFLD lli,l,xwb                      ; ...Store it in LLI
                             001F                                                     ;
            007C 8F  BB      001F              PUSHR   #^M<R2,R3,R4,R5,R6>            ; Save registers
           56  2C AA 9E      0023              MOVAB   CNF+LLI$Z_NDC_RT(R10),R6; Point to 'running total' counters
66    1C   00  6E  00 2C     0027              MOVC5   #0,(SP),#0,#NDC$C_LENGTH,(R6) ; Zero the counters
           56  48 AA 9E      002D              MOVAB   CNF+LLI$Z_NDC_LZ(R10),R6; Point to 'last zeroed' counters
66    1C   00  6E  00 2C     0031              MOVC5   #0,(SP),#0,#NDC$C_LENGTH,(R6) ; Zero the counters
                 56  D4      0037              CLRL    R6                             ; No "old" CNF
              FFC4'  30      0039              BSBW    CNF$INSERT                     ; Try to put block into list
            007C 8F  BA      003C              POPR    #^M<R2,R3,R4,R5,R6>            ; Restore registers
                             0040                                                     ;
              08 50  E8      0040              BLBS    R0,100$                        ; If LBS, okay
              FFBA'  30      0043              BSBW    NET$RELEASE_NDCOU              ; Else, dec. reference to counter block
       50   0000'8F  3C      0046      90$:    MOVZWL  #SS$_CONNECFAIL,R0             ; Return general purpose error status
                 05          004B      100$:   RSB                                    ; Release utility buffer, return status
                             004C
                       004C      .SBTTL  NET$CREATE_MBX  - Create ACP mailbox
                       004C      .SBTTL  NET$KILL_MBX    - Delete ACP mailbox
                       004C      .SBTTL  NET$MBX_QIO     - Issue mailbox read
                       004C      ;++
                       004C      ;
                       004C      ;
                       004C      ; *** TBS ***
                       004C      ;
                       004C      ;--
                       004C      NET$CREATE_MBX::
     004E'CF      B4   004C              CLRW    MBX_RDCNT                 ; Init oustanding mailbox read count
                       0050              $CREMBX_S -                       ; Create mailbox
                       0050                      CHAN   = MBX_CHAN,-
                       0050                      MAXMSG = #MBX_MSG_LTH,-
                       0050                      BUFQUO = #<MBX_MSG_LTH*16>,-
                       0050                      LOGNAM = NET$GQ_MBX_NAME,-  ; mailbox's logical name
                       0050                      PROMSK = #0
       16 50      E9   0071              BLBC    R0,10$                    ; Br if error
                       0074              $GETCHN_S -                       ; Get mailbox unit number
                       0074                      CHAN   = MBX_CHAN,-
                       0074                      PRIBUF = EXIT_BUF
                  05   008A      10$:    RSB                               ; Return status in R0
                       008B
                       008B
                       008B
                       008B      NET$KILL_MBX::                            ; Delete channel to mailbox
                       008B              $DASSGN_S  CHAN = MBX_CHAN        ; do it
                  05   0097              RSB
                       0098
                       0098
                       0098      NET$MBX_QIO::                                     ; Post read to mailbox
                       0098      ;
                       0098      ;
                       0098      ;  This routine puts a read out on the mailbox for process termination and
                       0098      ;  inbound connect notifications.
                       0098      ;
                       0098      ;
                       0098              $QIO_S  CHAN     = MBX_CHAN,-
                       0098                      FUNC     = S^#IO$_READVBLK,-
                       0098                      EFN      = #NET$C_EFN_ASYN,-
                       0098                      ASTADR   = NET$SET_MBX_AST,-
                       0098                      IOSB     = MBX_IOSB,-
                       0098                      P1       = EXIT_MSG,-
                       0098                      P2       = #MBX_MSG_LTH
       04 50      E8   00C1              BLBS    R0,10$                    ; Br unless error
                       00C4              BUG_CHECK       ACPMBFAIL,FATAL           ;!arrgh
                  05   00C8      10$:    RSB                                       ; return
```

```
                             00C9   .SBTTL  NET$SET_MBX_AST - Process mailbox AST
                             00C9   ;++
                             00C9   ;
                             00C9   ;
                             00C9   ;--
                             00C9   NET$SET_MBX_AST::
                       003C  00C9           .WORD   ^M<R2,R3,R4,R5>
                             00CB
        50   00F6'CF    9E   00CB           MOVAB   NET$GQ_WQE_MBX,R0          ; Get base of mailbox WQE
                14 A0    D5   00D0           TSTL    WQE$L_PM2(R0)              ; Is it active ?
                   0D    12   00D3           BNEQ    10$                        ; If NEQ then active, there's a bug
     10 A0    04 AC    D0   00D5           MOVL    4(AP),WQE$L_PM1(R0)        ; Get the AST parameter
        14 A0    01    CE   00DA           MNEGL   #1,WQE$L_PM2(R0)           ; Mark WQE busy
                FF1F'    30   00DE           BSBW    WQE$INSQUE                 ; Queue the WQE
                         04   00E1           RET                                ; Done
                             00E2
                             00E2   10$:    BUG_CHECK  NETNOSTATE,FATAL         ; Signal the bug
                             00E6
                             00E6   MBX_ACTION:                                 ; Enter upon WQE dispatch
                14 A5    D4   00E6           CLRL    WQE$L_PM2(R5)              ; Mark WQE idle
        EE'AF    00    FB   00E9           CALLS   #0,B^NET$MBX_AST           ; Call the mailbox processor
                         05   00ED           RSB
                             00EE   ;+
                             00EE   ;
                             00EE   ; NET$MBX_AST - THIS ROUTINE SERVICES PROCESS TERMINATIONS
                             00EE   ;               AND INBOUND CONNECT NOTIFICATIONS
                             00EE   ;
                             00EE   ;-
                       0000  00EE   NET$MBX_AST::   .WORD   0                  ; Entry point
       00'   0050'CF    B1   00F0           CMPW    MBX_IOSB,S^#SS$_ABORT      ; Was the i/o cancelled?
                   0E    13   00F5           BEQL    5$                         ; If so, assume mailbox going away
   0000'8F   0050'CF    B1   00F7           CMPW    MBX_IOSB,#SS$_CANCEL       ; Try this code, too
                   05    13   00FE           BEQL    5$                         ; If NEQ proceed
                   04    10   0100           BSBB    10$                        ; Dispatch
                 FF93    30   0102           BSBW    NET$MBX_QIO                ; Put out another read
                         04   0105   5$:     RET                                ; Done
                             0106           ;
                             0106           ;       Dispatch
                             0106           ;
        50   0054'CF    D0   0106   10$:    MOVL    MBX_PID,R0                 ; Get EPID returned by MBX driver
      00000000'GF    16   010B           JSB     G^EXE$EPID_TO_IPID         ; Convert to internal PID
     0054'CF    50    D0   0111           MOVL    R0,MBX_PID                 ; Use the IPID for later processing
        56   0000'CF    D0   0116           MOVL    NET$GL_NET_UCB,R6          ; Point to our NET channel's UCB
        5B   0058'CF    9E   011B           MOVAB   EXIT_ID,R11                ; Get address of mbx message
             56    8B    B0   0120           MOVW    (R11)+,R6                  ; Get message type
             59    8B    B0   0123           MOVW    (R11)+,R9                  ; Get unit number
             5A    8B    9A   0126           MOVZBL  (R11)+,R10                 ; Get device name count value
        00    6B    5A    2D   0129           CMPC5   R10,(R11),#0,-             ; X.25 mailbox message?
   007B'DF   0077'CF         012D                   X25_DEV_NAME,@X25_DEV_NAME+4
                   70    13   0133           BEQL    20$                        ; Branch if so
             5B    5A    C0   0135           ADDL    R10,R11                    ; Get pointer to mbx "data"
                             0138           $DISPATCH  TYPE=W,R6,-             ; Dispatch on mailbox msg type
                             0138           <-
                             0138             <MSG$_DELPROC,  DELPROC>,-         ; Process termination
                             0138             <MSG$_CONNECT,  CONNECT>,-         ; Inbound connect
                             0138             <MSG$_PATHLOST, NET$DRV_CANCEL>,-; I/O channel cancelled
                             0138           >
```

```
                      05  01A4          RSB                                 ; Ignore the message
                          01A5
                          01A5      ;
                          01A5      ;   Dispatch on X.25 mailbox message
                          01A5      ;
    5B   006C'CF    9E    01A5  20$:    MOVAB   EXIT_ID+20,R11              ; Point to "data"
    5A   0052'CF    3C    01AA          MOVZWL  MBX_LEN,R10                 ; Get length of mailbox message
         5A    14   C2    01AF          SUBL    #20,R10                     ; Subtract out overhead
                          01B2          $DISPATCH  TYPE=W,R6,-              ; Dispatch on mailbox msg type
                          01B2          <-
                          01B2            <MSG$_CONNECT,  NET$DLL_X25_CALL>,-    ; Incoming X.25 call
                          01B2            <MSG$_RESET,    NET$DLL_X25_RESET>,-   ; X.25 circuit reset
                          01B2          >
                      05  01D6          RSB                                 ; Ignore the message
```

```
                            01D7  ;
                            01D7  ;  Connect initiate message received
                            01D7  ;
                            01D7  ;
                            01D7  ;         The mailbox data consists of the address of the NETDRIVER update
                            01D7  ;         routine and the address of the XWB.
                            01D7  ;
                            01D7  CONNECT:
        56   04 AB     D0   01D7            MOVL    4(R11),R6                       ; Get XWB address
           50    01    D0   01DB            MOVL    #1,R0                           ; "Success" flag to NET$DELIVER_CI
               01FB    30   01DE            BSBW    NET$DELIVER_CI                  ; Deliver the inbound connect to a user
                       05   01E1            RSB
                            01E2
                            01E2  ;
                            01E2  ;  Handle network process termination
                            01E2  ;
                            01E2  DELPROC:
                            01E2            ;
                            01E2            ;  Notify netdriver of process exit
                            01E2            ;
           52    04    D0   01E2            MOVL    #NET$C_DR_NOBJ,R2                       ; Assume can't find .com file
                            01E5            CLRBIT  #STS$V_INHIB_MSG,EXIT_CODE              ; Ignore INHIB_MSG flag
005C'CF   00000000'8F  D1   01EB            CMPL    #RMS$_FNF,EXIT_CODE                     ; file-not-found?
                 11    13   01F4            BEQL    35$                                     ; if EQL yes
           52    26    D0   01F6            MOVL    #NET$C_DR_EXIT,R2                       ; assume just an exit
005C'CF   00000000'8F  D1   01F9            CMPL    #LGI$_INVPWD,EXIT_CODE                  ; was it an access problem?
                 03    12   0202            BNEQ    35$                                     ; if NEQ, no
           52    22    D0   0204            MOVL    #NET$C_DR_ACCESS,R2                     ; say so
        51   0054'CF   D0   0207  35$:      MOVL    MBX_PID,R1                              ; PID of exiting process
               005E    30   020C            BSBW    NET$SERVER_FAIL                         ; Notify NETSERVER server gone
                            020F            ;
                            020F            ;  Remove the process from the SPI database
                            020F            ;
        5B   0000'CF   D0   020F            MOVL    NET$GL_CNR_SPI,R11                      ; Get root of SPI database
                 5A    D4   0214            CLRL    R10                                     ; Start at beginning of list
        58   0054'CF   D0   0216            MOVL    MBX_PID,R8                              ; Get process PID
                            021B            $SEARCH eql,spi,l,pid                           ; Find database entry
              18 50    E9   0228            BLBC    R0,50$                                  ; Branch if not found
                            022B            $GETFLD spi,l,irp                               ; Waiting DECLSERV IRP?
              04 50    E9   0236            BLBC    R0,40$                                  ; Branch if no IRP waiting
                            0239            BUG_CHECK NETNOSTATE,FATAL                      ; Should never have IRP
                            023D                                                            ; waiting, if process deleted
              FDC0'    30   023D  40$:      BSBW    CNF$DELETE                              ; Mark the CNF entry deleted
              FDBD'    30   0240            BSBW    CNF$PURGE                               ; Delete all marked CNFs
                            0243  50$:      ;
                            0243            ;  Notify DLE module of process termination.
                            0243            ;
        58   0054'CF   D0   0243            MOVL    MBX_PID,R8                              ; Setup the PID
              FDB5'    30   0248            BSBW    DLE$PRC_EXIT                            ; Inform DLE of process exit
                 05         024B            RSB
```

```
                          024C  .SBTTL  NET$CONNECT_FAIL - Notify NETDRIVER of failed link
                          024C  ;+
                          024C  ;
                          024C  ; An attempt to confirm a logical link has failed.  Notify NETDRIVER so that
                          024C  ; it can verify the user's access to the link and then notify the remote end of
                          024C  ; the link that the link is being broken and why.
                          024C  ;
                          024C  ;
                          024C  ; INPUTS:        R3      Local logical link number (0 implies connect initiate)
                          024C  ;                R2      Reason code to be sent in the disconnect message
                          024C  ;                R1      User's PID
                          024C  ;
                          024C  ; OUTPUTS:       R5-R0   Clobbered
                          024C  ;
                          024C  ;                All other registers are preserved
                          024C  ;
                          024C  ;-
                          024C  NET$CONNECT_FAIL::                                      ; A connect attempt has failed
0014'CF   0000'CF   D0    024C          MOVL    NET$GL_NET_UCB,NET_L_UCB                ; Use the ACP's UCB
     000C'CF   53   3C    0253          MOVZWL  R3,NET_L_LNK                            ; Specify link number
               12   13    0258          BEQL    10$                                     ; If EQL then connect initiate
     0008'CF   52   D0    025A          MOVL    R2,NET_L_REASON                         ; Specify disconnect reason
     0004'CF   51   D0    025F          MOVL    R1,NET_L_PID                            ; Specify user's PID
     0000'CF   01   D0    0264          MOVL    #NETUPD$_ABORT,NET_L_FCT                ; Specify "link terminated"
             067F   30    0269          BSBW    TELL_DRV                                ; Notify NETDRIVER
                    05    026C  10$:    RSB
```

```
                              026D  .SBTTL  NET$SERVER_FAIL - Notify NETDRIVER of terminated server
                              026D  ;+
                              026D  ;
                              026D  ; A server process (or NETSERVER session) has terminated.  Notify NETDRIVER so
                              026D  ; that it can break all links that might still be pending for that process.
                              026D  ; This handles the case where the process was unable to confirm the link due
                              026D  ; to some error.
                              026D  ;
                              026D  ;
                              026D  ; INPUTS:         R2      Reason code to be sent in the disconnect message
                              026D  ;                 R1      User's PID
                              026D  ;
                              026D  ; OUTPUTS:        R5-R0   Clobbered
                              026D  ;
                              026D  ;                 All other registers are preserved
                              026D  ;
                              026D  ;-
                              026D  NET$SERVER_FAIL::                                       ; A server has terminated
      0014'CF   0000'CF   D0  026D          MOVL    NET$GL_NET_UCB,NET_L_UCB                ; Use the ACP's UCB
           0008'CF   52   D0  0274          MOVL    R2,NET_L_REASON                         ; Specify disconnect reason
           0004'CF   51   D0  0279          MOVL    R1,NET_L_PID                            ; Specify user's PID
           0000'CF   03   D0  027E          MOVL    #NETUPD$_EXIT,NET_L_FCT                 ; Specity "process exit"
                0665  30  0283          BSBW    TELL_DRV                                ; Notify NETDRIVER
                      05  0286          RSB
```

```
                         0287  .SBTTL  NET$SCAN_FOR_ZNA - Send pending connects to declared object
                         0287  ;+
                         0287  ;
                         0287  ; This routine is called when a task name or object is declared by a user.
                         0287  ; The function is to scan the XWB list for links in the Connect Initiate
                         0287  ; state which are intended for the object with the specified ZNA and to build
                         0287  ; a NCB which is given to NETDRIVER to be put in the declared name's mailbox.
                         0287  ;
                         0287  ; INPUTS:        R7,R8 = Descriptor of object ZNA being declared
                         0287  ;
                         0287  ; OUTPUTS:       None
                         0287  ;
                         0287  ;-
                         0287  NET$SCAN_FOR_ZNA::                                    ; Find unclaimed XWBs which match ZNA
             59   57  7D 0287              MOVQ    R7,R9                          ; Make copy of ZNA descriptor
        56     0000'CF D0 028A              MOVL    NET$GL_PTR_VCB,R6              ; Get RCB pointer
          56    24 A6  D0 028F              MOVL    RCB$L_PTR_LTB(R6),R6           ; Get LTB pointer
          55    14 A6  9E 0293              MOVAB   LTB$L_SLOTS+4(R6),R5           ; Point to first XWB (skip slot 0)
                FD 85  E8 0297   10$:       BLBS    (R5)+,10$                      ; If LBS then pointer is invalid
          56    FC A5  D0 029A              MOVL    -4(R5),R6                      ; Get the XWB address
                   29  13 029E              BEQL    30$                            ; If EQL then done
                   03  91 02A0              CMPB    #XWB$C_STA_CIR,-               ; In connect initiate state?
                1E A6     02A2                          XWB$B_STA(R6)              ;
                   F1  12 02A4              BNEQ    10$                            ; If NEQ then keep looking
        51     00A5 C6  9E 02A6              MOVAB   XWB$T_LPRNAM(R6),R1            ; Setup for subroutine call
               0660 8F  BB 02AB              PUSHR   #^M<R5,R6,R9,R10>              ; Save important regs
                  05F1  30 02AF              BSBW    GET_PR_ZNA                     ; Get ZNA string from LRPNAM
                0E 50  E9 02B2              BLBC    R0,20$                         ; If LBC then field is not valid
  6A   59   00   68 57  2D 02B5              CMPC5   R7,(R8),#0,R9,(R10)            ; Are they the same?
                   06  12 02BB              BNEQ    20$                            ; If NEQ keep looking
          50       01  D0 02BD              MOVL    #1,R0                          ; "Success" flag to NET$DELIVER_CI
                  0119  30 02C0              BSBW    NET$DELIVER_CI                 ; Build NCB, pass to user in mailbox
               0660 8F  BA 02C3   20$:       POPR    #^M<R5,R6,R9,R10>              ; Restore regs
                   CE  11 02C7              BRB     10$                            ; Keep looking
                       05 02C9   30$:       RSB                                    ; Done
```

```
                              02CA  .SBTTL  NET$RESEND_SERVER - Re-send initial connect to server
                              02CA  ;+
                              02CA  ;
                              02CA  ; This routine is called when a server process declares that it is waiting
                              02CA  ; for an incoming connect.  The XWB list is scanned for links in the CI
                              02CA  ; state looking to see if the initial connect which started the process
                              02CA  ; is still pending.  If so, then re-send the NCB to the server process
                              02CA  ; so that it will be executed.
                              02CA  ;
                              02CA  ; INPUTS:          R8 = PID of server process
                              02CA  ;
                              02CA  ; OUTPUTS:         None
                              02CA  ;
                              02CA  ;-
                              02CA  NET$RESEND_SERVER::                                   ; Find unclaimed XWBs for server process
         56   0000'CF   D0    02CA          MOVL    NET$GL_PTR_VCB,R6                   ; Get RCB pointer
          56    24 A6   D0    02CF          MOVL    RCB$L_PTR_LTB(R6),R6                ; Get LTB pointer
          55    14 A6   9E    02D3          MOVAB   LTB$L_SLOTS+4(R6),R5                ; Point to first XWB (skip slot 0)
                FD 85   E8    02D7  10$:    BLBS    (R5)+,10$                           ; If LBS then pointer is invalid
          56    FC A5   D0    02DA          MOVL    -4(R5),R6                           ; Get the XWB address
                   1C   13    02DE          BEQL    30$                                 ; If EQL then done
                   03   91    02E0          CMPB    #XWB$C_STA_CIR,-                    ; In connect initiate state?
                1E A6         02E2                  XWB$B_STA(R6)                       ;
                   F1   12    02E4          BNEQ    10$                                 ; If NEQ then keep looking
         34 A6     58   D1    02E6          CMPL    R8,XWB$L_PID(R6)                    ; Intended for this process?
                   EB   12    02EA          BNEQ    10$                                 ; If NEQ keep looking
              0160 8F   BB    02EC          PUSHR   #^M<R5,R6,R8>                       ; Save registers
             50    01   D0    02F0          MOVL    #1,R0                               ; "Success" flag to NET$DELIVER_CI
                 00E6   30    02F3          BSBW    NET$DELIVER_CI                      ; Build NCB, satisfy DECLSERV request
              0160 8F   BA    02F6          POPR    #^M<R5,R6,R8>                       ; Restore registers
                   DB   11    02FA          BRB     10$                                 ; Keep looking
                        05    02FC  30$:    RSB                                         ; Done
```

```
                          02FD  .SBTTL  NET$STARTUP_OBJ      - Startup privileged process
                          02FD  .SBTTL  NET$STARTUP_OBJ_NAM  - Startup process by name
                          02FD  ;+
                          02FD  ;
                          02FD  ; Startup a privileged object process if it is not already running.  This is
                          02FD  ; used to create EVL for event logging and NML for down-line loading or
                          02FD  ; up-line dumping.
                          02FD  ;
                          02FD  ; Inputs:
                          02FD  ;
                          02FD  ;         R8    = Object number to start (If NET$STARTUP_OBJ)
                          02FD  ;         R7/R8 = Object name to start (If NET$STARTUP_OBJ_NAM)
                          02FD  ;
                          02FD  ;         R2,R3 = Descriptor of string to be passed as SYS$NET to process
                          02FD  ;         R4,R5 = Descriptor of string to be used as process name
                          02FD  ;                 If =0 then use the object's name as the process name
                          02FD  ;
                          02FD  ; Outputs:
                          02FD  ;
                          02FD  ;         R1    PID if process has been created
                          02FD  ;         R0    Status
                          02FD  ;-
                          02FD            .ENABL  LSB
                          02FD
                          02FD  NET$STARTUP_OBJ_NAM::
        0F80 8F       BB  02FD            PUSHR   #^M<R7,R8,R9,R10,R11>    ; Save registers
    5B    0000'CF     D0  0301            MOVL    NET$GL_CNR_OBI,R11       ; Point to OBI database
              5A      D4  0306            CLRL    R10                      ; and start at beginning of list
                          0308            $SEARCH eql,obi,l,nam            ; Search for specified object
              18      11  0315            BRB     1$                       ; Join common code
                          0317
                          0317  NET$STARTUP_OBJ::                          ; Startup privileged process
        0F80 8F       BB  0317            PUSHR   #^M<R7,R8,R9,R10,R11>    ; Save registers
    5B    0000'CF     D0  031B            MOVL    NET$GL_CNR_OBI,R11       ; Point to OBI database
              5A      D4  0320            CLRL    R10                      ; and start at beginning of list
                          0322            $SEARCH eql,obi,l,num            ; Search for specified object
              51      D4  032F  1$:       CLRL    R1                       ; Clear PID
    0024'CF   52      7D  0331            MOVQ    R2,NET_Q_NCB             ; Store descriptor of SYS$NET string
           1C 50      E9  0336            BLBC    R0,2$                    ; Skip if not defined as object
                          0339            ;
                          0339            ; If object has already declared itself, then it is running
                          0339            ;
                          0339            $GETFLD obi,l,ucb                ; If UCB NE 0, it has declared itself
           0E 50      E8  0344            BLBS    R0,2$                    ; If declared, then its already running
                          0347            ;
                          0347            ; If not, get the access control string and process name
                          0347            ;
                          0347            $GETFLD obi,s,sfi                ; Get the process file name
           03 50      E8  0352            BLBS    R0,5$                    ; Skip if specified
            007F      31  0355  2$:       BRW     80$                      ; Return with status in R0
    0034'CF   57      7D  0358  5$:       MOVQ    R7,NET_Q_TSK             ; Save the descriptor
    002C'CF   54      7D  035D            MOVQ    R4,NET_Q_PRC             ; Setup process name
              10      12  0362            BNEQ    10$                      ; If NEQ then name is non-null
                          0364            $GETFLD obi,s,nam                ; Else get object name
    002C'CF   57      7D  036F            MOVQ    R7,NET_Q_PRC             ; Use as process name
    003C'CF   05      D0  0374  10$:      MOVL    #DET_C_ACC,NET_Q_ACC     ; Setup descriptor of access control
 0040'CF   0044'CF    9E  0379            MOVAB   DET_AB_ACC,NET_Q_ACC+4   ; data used for create detached,
                          0380                                                  ; privileged processes.
                          0380              ;
                          0380              ;  Start the process with privileges
                          0380              ;
                          0380              $CREPRC_S       -                   ; create a process
                          0380                 INPUT  = NET_Q_TSK,-             ; Network .COM filename
                          0380                 OUTPUT = NET_Q_ACC,-             ; Network access control string
                          0380                 ERROR  = NET_Q_NCB,-             ; SYS$NET logical name string
                          0380                 PRCNAM = NET_Q_PRC,-             ; Process name
                          0380                 IMAGE  = NET_Q_IMAGE,-           ; Image (LOGINOUT) to run first
                          0380                 PIDADR = NET_L_PID,-             ; Place to store process id
                          0380                 MBXUNT = MBX_UNIT,-              ; MBX for termination
                          0380                 BASPRI = G^SYS$GB_DEFPRI,-       ; Priority
                          0380                 UIC    = #<^O1@16+^O4>,-         ; UIC is [1,4]
                          0380                 STSFLG = <#STS_M_NETLOG!-        ; Network login parameters (IN,OUT,ERR)
                          0380                           STS_M_NOAUTH!-         ; Use caller's privs/quotas/etc.
                          0380                           STS_M_NOACNT>          ; Do not add any accounting records
           18 50    E9    03BC              BLBC    R0,80$                      ; If LBC then failed
   50    0004'CF    D0    03BF              MOVL    NET_L_PID,R0                ; Get the EPID returned by CREPRC
     00000000'GF    16    03C4              JSB     G^EXE$EPID_TO_IPID          ; Convert to internal PID format
  0004'CF    50     D0    03CA              MOVL    R0,NET_L_PID                ; Use internal format of PID
  51    0004'CF     D0    03CF              MOVL    NET_L_PID,R1                ; Return the PID to caller
       50    00'    D0    03D4              MOVL    S^#SS$_NORMAL,R0            ; Success
         0F80 8F    BA    03D7      80$:    POPR    #^M<R7,R8,R9,R10,R11>       ; Restore registers
              05          03DB              RSB
                          03DC
                          03DC              .DSABL  LSB
```

```
                               03DC  .SBTTL  NET$DELIVER_CI  - Process and Deliver Inbound Connect
                               03DC  ;++
                               03DC  ;
                               03DC  ;  A non-zero destination object number indicates that NETACP must fetch the
                               03DC  ;  name of the .COM file from the OBJ block - using 'SYS$SYSROOT:[SYSEXE]' as
                               03DC  ;  the default directory.  A zero destination object number indicates that the
                               03DC  ;  .COM file name is the same as the destination taskname - the default login
                               03DC  ;  directory account is assumed to contain the taskname.COM.
                               03DC  ;
                               03DC  ;           \update this to include tasks with a file i.d.\;!
                               03DC  ;
                               03DC  ;  If the incoming USER,PSW,ACCT strings are all null, then the default
                               03DC  ;  inbound access control for the specified object (or task) are used (these
                               03DC  ;  strings may also be null).  This allows a DECnet-VAX node to serve as a
                               03DC  ;  convenient host particularly for RSX-11S.
                               03DC  ;
                               03DC  ;  This routines determines whether the connect is to be handed to a task
                               03DC  ;  which has declared a name or an object type.
                               03DC  ;
                               03DC  ;
                               03DC  ;  INPUTS:         R11     LLI CNR address (if low bit set in R0)
                               03DC  ;                  R10     LLI CNF address (if low bit set in R0)
                               03DC  ;                  R6      XWB address
                               03DC  ;                  R0      Low bit set     =>  deliver connect notification
                               03DC  ;                          Low bit clear   =>  tell NETDRIVER that resource error
                               03DC  ;                                              occurred
                               03DC  ;
                               03DC  ;  OUTPUTS:        R11,R10,R6 are preserved.
                               03DC  ;
                               03DC  ;                  All other registers are clobbered.
                               03DC  ;
                               03DC  ;
                               03DC  ; SIDE EFFECTS: Process created if needed, image started
                               03DC  ;
                               03DC  ;--
                               03DC
                               03DC  ;
                               03DC  ; Define scratch storage
                               03DC  ;
                               03DC
                   0000000C    03DC  ACC =  12                                       ; Composite access strings
                   000000C8    03DC  PRC = 200                                       ; Process name
                   0000012C    03DC  TSK = 300                                       ; Image to run
                   000003E8    03DC  CONN_SPACE = 1000                               ; Size of scratch storage
                               03DC
                               03DC  NET$DELIVER_CI:
            0018'CF      D4    03DC          CLRL    PTR_NCB_BUF                     ; No NCB buffer yet
            001C'CF      D4    03E0          CLRL    PTR_CON_BUF                     ; No scratch buffer yet
                               03E4          ;
                               03E4          ;     Initialize parameters for call to NETDRIVER
                               03E4          ;
     000C'CF    3E A6    3C    03E4          MOVZWL  XWB$W_LOCLNK(R6),NET_L_LNK              ; Setup logical link address
       0000'CF    01     9A    03EA          MOVZBL  #NETUPD$_ABORT,   NET_L_FCT             ; Assume process couldn't start
   0014'CF    0000'CF    D0    03EF          MOVL    NET$GL_NET_UCB,   NET_L_UCB             ; Default is our UCB
       0020'CF    5A     D0    03F6          MOVL    R10,              NET_A_LLI             ; Save LLI pointer
            0004'CF      D4    03FB          CLRL                      NET_L_PID             ; No PID yet
            0010'CF      D4    03FF          CLRL                      NET_A_NCB             ; No NCB yet
```

```
                        0403           ;
                        0403           ;
                        0403           ;
          57 50    E9   0403           BLBC    R0,3$                       ; If LBC, resource error encountered
                        0406                                               ; by caller
     51  03E8 8F   3C   0406           MOVZWL  #CONN_SPACE,R1              ; Set size of scratch buffer
             FBF2'  30  040B           BSBW    NET$ALLOCATE                ; Allocate a scratch buffer
          4C 50    E9   040E           BLBC    R0,3$                       ; Br if allocation failure, notify
                        0411                                               ;  driver
    001C'CF    52  D0   0411           MOVL    R2,PTR_CON_BUF              ; Save address for deallocation
                        0416           ;
                        0416           ;       Initialize descriptors and data for process creation
                        0416           ;
      53    0C A2  9E   0416           MOVAB   ACC(R2),R3                  ; Get ACC address
    0040'CF    53  D0   041A           MOVL    R3,NET_Q_ACC+4              ; Store it
    003C'CF    53  CE   041F           MNEGL   R3,NET_Q_ACC                ; Bias ACC size
     53   00C8 C2  9E   0424           MOVAB   PRC(R2),R3                  ; Get PRC address
    0030'CF    53  D0   0429           MOVL    R3,NET_Q_PRC+4              ; Store it
    002C'CF    53  CE   042E           MNEGL   R3,NET_Q_PRC                ; Bias PRC size
     53   012C C2  9E   0433           MOVAB   TSK(R2),R3                  ; Get TSK address
    0038'CF    53  D0   0438           MOVL    R3,NET_Q_TSK+4              ; Store it
    0034'CF    53  CE   043D           MNEGL   R3,NET_Q_TSK                ; Bias TSK size
                        0442           ;
                        0442           ;       Set default values
                        0442           ;
     50   0000'CF  D0   0442           MOVL    NET$GL_PTR_VCB,R0                  ; Point to RCB
  0049'CF   67 A0  90   0447           MOVB    RCB$B_ECL_DPX(R0),OBI_B_PRX        ; Set default OBI proxy access
    004A'CF    03  90   044D           MOVB    #NMA$C_ACES_BOTH, INT_B_PRX        ; Set default internal proxy
                        0452                                                      ; access state
                        0452           ;
                        0452           ;       Allocate scratch buffer from nonpaged pool for NCB
                        0452           ;
     51   007B 8F  3C   0452           MOVZWL  #NET$C_MAX_NCB+13,R1               ; Length of buffer for an NCB
             FBA6'  30  0457           BSBW    NET$ALONPAGED                      ; Allocate the buffer
          17 50    E8   045A           BLBS    R0,5$                              ; If LBS then block allocated
                        045D           ;
                        045D           ;     Tell NETDRIVER about resource error
                        045D           ;
    0008'CF    01  9A   045D   3$:     MOVZBL  #NET$C_DR_RSU,NET_L_REASON         ; Reason is "resource error"
     50   0000'CF  D0   0462           MOVL    NET$GL_PTR_VCB,R0                  ; Get RCB pointer
                        0467           BUMP    W,RCB$W_CNT_XRE(R0)                ; Account for resource error
             34    11   0472           BRB     10$                                ; Continue
                        0474           ;
                        0474           ;   Build the NCB and locate the process to accept it
                        0474           ;
    0018'CF    52  D0   0474   5$:     MOVL    R2,PTR_NCB_BUF                     ; Save for deallocation
      53    0D A2  9E   0479           MOVAB   13(R2),R3                          ; Get address of string, leave
                        047D                                                      ; room for count and buf header
    0028'CF    53  D0   047D           MOVL    R3,NET_Q_NCB+4                     ; Store it
    0024'CF    53  CE   0482           MNEGL   R3,NET_Q_NCB                       ; Bias NCB size
             0032  30   0487           BSBW    BUILD_NCB                          ; Build the NCB
          1B 50    E9   048A           BLBC    R0,10$                             ; If LBC then error
0000006E 8F  0024'CF  D1   048D        CMPL    NET_Q_NCB,#NET$C_MAX_NCB           ; Make sure we didn't write
                        0496                                                      ; past end of buffer
                        0496           ASSUME  NET$C_MAX_NCB  LE  255             ; Must allow counted string fmt
     50   0028'CF  D0   0496           MOVL    NET_Q_NCB+4,R0                     ; Get ptr to NCB
     70   0024'CF  90   049B           MOVB    NET_Q_NCB,-(R0)                    ; Enter count field and
    0010'CF     50   D0  04A0            MOVL    R0,NET_A_NCB            ; save its address in case NCB
                         04A5                                            ; is to be passed to NETDRIVER
                         04A5                                            ; for a declared name
           00B2      30  04A5            BSBW    GET_PROC                ; Find/create process to
                         04A8                                            ; receive the connect
           0440      30  04A8     10$:   BSBW    TELL_DRV                ; Tell driver about connect
50     0018'CF       D0  04AB            MOVL    PTR_NCB_BUF,R0          ; Address of buffer
          FB4D'      30  04B0            BSBW    NET$DEALLOCATE          ; Deallocate the buffer
50     001C'CF       D0  04B3            MOVL    PTR_CON_BUF,R0          ; Address of scratch buffer
          FB45'      30  04B8            BSBW    NET$DEALLOCATE          ; Deallocate scratch storage
                     05  04BB            RSB                             ; Done
                         04BC
```

```
                           04BC  .SBTTL  BUILD_NCB        - Build NCB for incoming connect
                           04BC  ;+
                           04BC  ;
                           04BC  ; This routine builds the NCB string for the connect, to be later
                           04BC  ; given to the destination process (in any number of different ways).
                           04BC  ;
                           04BC  ; Inputs:
                           04BC  ;
                           04BC  ;       R6 = XWB address
                           04BC  ;       NET_Q_NCB = Descriptor of scratch space for NCB
                           04BC  ;
                           04BC  ; Outputs:
                           04BC  ;
                           04BC  ;       R0 = status code
                           04BC  ;       NET_Q_NCB = Descriptor of resultant NCB
                           04BC  ;-
                           04BC  BUILD_NCB:                                      ; Build the NCB
                           04BC          ;
                           04BC          ;       Enter 'nodename::'
                           04BC          ;
 53    0028'CF     D0      04BC          MOVL    NET_Q_NCB+4,R3                  ; Get output buffer pointer
 5B    0000'CF     D0      04C1          MOVL    NET$GL_CNR_NDI,R11              ; Get root for search
            5A     D4      04C6          CLRL    R10                             ; Indicate no NDI yet
   58    3A A6     3C      04C8          MOVZWL  XWB$W_REMNOD(R6),R8             ; Get remote node address
                           04CC          $SEARCH eql,ndi,l,tad                   ; Find NDI with matching address
         18 50     E9      04D9          BLBC    R0,10$                          ; If LBC none, use node address
                           04DC          $GETFLD ndi,s,nna                       ; Get the node name
         0A 50     E9      04E7          BLBC    R0,10$                          ; Invalid if LBC
            57     95      04EA          TSTB    R7                              ; Is name null ?
            06     13      04EC          BEQL    10$                             ; If EQL use node address
                           04EE          ;
                           04EE          ;       Enter ASCII nodename
                           04EE          ;
 63    68   57     28      04EE          MOVC3   R7,(R8),(R3)                    ; Move node name
            07     11      04F2          BRB     20$
                           04F4          ;
                           04F4          ;       Enter node address converted to ASCII
                           04F4          ;
   50    3A A6     3C      04F4  10$:    MOVZWL  XWB$W_REMNOD(R6),R0             ; Get node address
         FB05'     30      04F8          BSBW    NET$BIN2ASC                     ; Move after conversion to ASCII
 83    3A3A 8F     B0      04FB  20$:    MOVW    #^A'::',(R3)+                   ; Move delimiter
                           0500          ;
                           0500          ;       Enter taskname
                           0500          ;
      83    22     90      0500          MOVB    #^A'"',(R3)+                    ; Enter delimiter
 50    00BA C6     9A      0503          MOVZBL  XWB$T_RPRNAM+1(R6),R0           ; Get object number
            08     13      0508          BEQL    30$                             ; If EQL then use taskname
         FAF3'     30      050A          BSBW    NET$BIN2ASC                     ; Else convert to ASCII and move
      83    3D     90      050D          MOVB    #^A'=',(R3)+                    ; Enter delimiter
            19     11      0510          BRB     50$                             ; Continue
                           0512          ;
                           0512          ;       Enter 0=taskname
                           0512          ;
 83    3D30 8F     B0      0512  30$:    MOVW    #^A'0=',(R3)+                   ; Enter 0=
 51    00B9 C6     9E      0517          MOVAB   XWB$T_RPRNAM(R6),R1             ; Point to process name field
            53     DD      051C          PUSHL   R3                              ; Save pointer
          0378     30      051E          BSBW    GET_PR_NAM                      ; Move the name text
```

```
                    53 8ED0  0521        POPL    R3                        ; Recover pointer
                 32 50   E9  0524        BLBC    R0,60$                    ; If LBC then illegal name format
          63  68    57   28  0527        MOVC3   R7,(R8),(R3)              ; Enter taskname
                             052B   ;
                             052B   ;        Enter remainder of NCB
                             052B   ;
              83    2F   90  052B  50$:  MOVB    #^A'/',(R3)+              ; Enter delimiter
          83  000C'CF    B0  052E        MOVW    NET_L_LNK,(R3)+           ; Enter local link number
           51    5B A6   9E  0533        MOVAB   XWB$B_DATA(R6),R1         ; Get address of counted data
              50    61   9A  0537        MOVZBL  (R1),R0                   ; Get its length
                    50   B6  053A        INCW    R0                        ; Include its count field
 63   11  00  61    50   2C  053C        MOVC5   R0,(R1),#0,#17,(R3)       ; Enter into fixed size field
          51  00A4  C6   9E  0542        MOVAB   XWB$B_LPRNAM(R6),R1       ; Address local task specifier
              50    81   9A  0547        MOVZBL  (R1)+,R0                  ; Get its length
          63  61    50   28  054A        MOVC3   R0,(R1),(R3)              ; Move it
              83    22   90  054E        MOVB    #^A'"',(R3)+              ; Enter terminator
        0024'CF     53   C0  0551        ADDL    R3,NET_Q_NCB              ; Update size in descriptor
              50    01   90  0556        MOVB    #1,R0                     ; Indicate success
                         05  0559  60$:  RSB
```

```
                          055A  .SBTTL  GET_PROC        - Locate process to accept connect
                          055A  ;+
                          055A  ;
                          055A  ; Find the OBI block associated with the local object.  If the OBI is
                          055A  ; for a declared name or object then pass the NCB to the declaring
                          055A  ; process's mailbox, otherwise create a process to receive the connect.
                          055A  ; If there is a server process waiting for more work, then tell the
                          055A  ; server process that it can have the connect request.
                          055A  ;
                          055A  ; Inputs:
                          055A  ;
                          055A  ;       R6 = XWB address
                          055A  ;
                          055A  ;       Own storage
                          055A  ;
                          055A  ; Outputs:
                          055A  ;
                          055A  ;       None
                          055A  ;-
                          055A  GET_PROC:                                       ; Get process to accept the connect
  5B    0000'CF    D0     055A          MOVL    NET$GL_CNR_OBI,R11              ; Set up OBI CNR
  51    00A5 C6    9E     055F          MOVAB   XWB$T_LPRNAM(R6),R1             ; Address local task specifier
           033C    30     0564          BSBW    GET_PR_ZNA                      ; Get its ZNA field
          2B 50    E9     0567          BLBC    R0,10$                          ; If LBC then format error
                          056A          ;
                          056A          ;
                          056A          ;       Find the OBI CNF
                          056A          ;
                          056A          ;
             04    3C     056A          MOVZWL  #NET$C_DR_NOBJ,-                ; Assume failure due to unknown object
        0008'CF           056C                  NET_L_REASON                    ;
             5A    D4     056F          CLRL    R10                             ; Indicate no current CNF
                          0571          $SEARCH eql,obi,s,zna                   ; Find OBI block with this CNF
          17 50    E8     057E          BLBS    R0,20$                          ; If LBS then CNF was found
             68    95     0581          TSTB    (R8)                            ; Is this a numbered object connect ?
             10    12     0583          BNEQ    10$                             ; If NEQ then no such object
  57    000B'CF    7D     0585          MOVQ    NET_Q_TASKZNA,R7                ; Else use default TASK  ZNA descriptor
     51     00     9A     058A          MOVZBL  S^#NFB$C_OP_EQL,R1              ; Specify match operator
             5A    D4     058D          CLRL    R10                             ; Start from head of list
          FA6E'    30     058F          BSBW    CNF$KEY_SEARCH                  ; Look for the CNF
          3A 50    E8     0592          BLBS    R0,25$                          ; If LBS then found, br to continue
           021B    31     0595  10$:    BRW     100$                            ; Complete with error
                          0598          ;
                          0598          ;
                          0598          ;       The OBI CNF has been found.  See if the object has been "declared"
                          0598          ;       If not, build the .COM file file i.d. and setup its descriptor.
                          0598          ;
                          0598          ;
                          0598  20$:    $GETFLD obi,l,ucb                       ; Get the associated UCB
          52 50    E9     05A3          BLBC    R0,30$                          ; If LBC then not declared name
     0014'CF  58   D0     05A6          MOVL    R8,NET_L_UCB                    ; Save the UCB pointer
                          05AB          $GETFLD obi,l,pid                       ; Get the declarer's EPID
          3F 50    E9     05B6          BLBC    R0,30$                          ; If LBC then treat as undeclared
          50  58   D0     05B9          MOVL    R8,R0                           ; Convert from EPID to IPID
      00000000'GF  16     05BC          JSB     G^EXE$EPID_TO_IPID              ; ...
     0004'CF  50   D0     05C2          MOVL    R0,NET_L_PID                    ; Save the PID
             02    9A     05C7          MOVZBL  #NETUPD$_CONNECT,-              ; Setup the function code
        0000'CF          05C9                   NET_L_FCT              ;
           01E4   31     05CC          BRW      100$                   ; Return to pass NCB to mailbox
                         05CF          ;
                         05CF          ;   The object is a named object which could not be found in the
                         05CF          ;   object database.  Use the requested object name to construct
                         05CF          ;   the name of the command procedure, rather than consulting the
                         05CF          ;   OBI entry (we are currently set to the "TASK" OBI).  If the
                         05CF          ;   object name starts with a "$", then the object is "reserved
                         05CF          ;   to DEC", and we get the command procedure from SYS$SYSTEM.
                         05CF          ;
  51    00A5 C6   9E     05CF  25$:    MOVAB    XWB$T_LPRNAM(R6),R1    ; Address local task specifier
           02C2   30     05D4          BSBW     GET_PR_NAM             ; Get its name
  53    0038'CF   D0     05D7          MOVL     NET_Q_TSK+4,R3         ; Get address of output buffer
     24     68    91     05DC          CMPB     (R8),#^A"$"            ; Does the name start with "$"?
             0C   12     05DF          BNEQ     28$                    ; If so,
             58   D6     05E1          INCL     R8                     ; Strip "$" off front of name
             57   D7     05E3          DECL     R7
        0032'CF   28     05E5          MOVC     NET_Q_SYSTEM,-         ; Prefix name with "SYS$SYSTEM:"
  63    0036'DF          05E9                   @NET_Q_SYSTEM+4,(R3)
  63    68   57   28     05ED  28$:    MOVC3    R7,(R8),(R3)           ; Move the name
  0034'CF    53   C0     05F1          ADDL     R3,NET_Q_TSK           ; Update filename size
             26   11     05F6          BRB      40$                    ; Continue
                         05F8          ;
                         05F8          ;   Build filespec of object command procedure
                         05F8          ;
             04   3C     05F8  30$:    MOVZWL   #NET$C_DR_NOBJ,-       ; Assume error
        0008'CF          05FA                   NET_L_REASON           ;
                         05FD          $GETFLD obi,s,sfi               ; Get parsed file id
          6B 50   E9     0608          BLBC     R0,55$                 ; If LBC then file id is invalid
  0034'CF    57   7D     060B          MOVQ     R7,NET_Q_TSK           ; Update filename descriptor
                         0610          ;
                         0610          ;
                         0610          ;   Create a process name.
                         0610          ;
                         0610          ;
                         0610          $GETFLD obi,s,nam               ; Get object name for prefix
          05 50   E8     061B          BLBS     R0,50$                 ; If LBS then name was found
  57    0000'CF   7D     061E  40$:    MOVQ     NET_Q_NETPREFIX,R7     ; Setup standard prefix descriptor
0030'DF   68   57   28   0623  50$:    MOVC3    R7,(R8),@NET_Q_PRC+4   ; Move the prefix
     83    5F 8F  90     0629          MOVB     #^A'_',(R3)+           ; Move the delimiter
  50    000C'CF   D0     062D          MOVL     NET_L_LNK,R0           ; Get the local link number
          F9CB'   30     0632          BSBW     NET$BIN2ASC            ; Convert to ascii and append as
                         0635                                          ; the suffix
  002C'CF    53   C0     0635          ADDL     R3,NET_Q_PRC           ; Done with process name
                         063A          ;
                         063A          ;   If the connect did not use format type 2, then don't attempt
                         063A          ;   a proxy login.
                         063A          ;
  02    00B9 C6   91     063A          CMPB     XWB$T_RPRNAM(R6),#2    ; Format type 2?
             05   13     063F          BEQL     51$                    ; Branch if so
  004A'CF    00   90     0641          MOVB     #NMA$C_ACES_NONE,INT_B_PRX ; Disallow proxy access
                         0646  51$:    ;
                         0646          ;
                         0646          ;   If no access control was specified, use default from OBI block
                         0646          ;
                         0646          ;
                         0646          $GETFLD obi,l,prx               ; Get proxy login state
```

```
         05 50    E9 0651            BLBC    R0,52$                  ; If LBC then none specified
    0049'CF  58    90 0654            MOVB    R8,OBI_B_PRX            ; Store it
 58     00C6 66    9E 0659  52$:      MOVAB   XWB$B_LOGIN(R6),R8      ; Get address of access info
          57  88    9A 065E            MOVZBL  (R8)+,R7                ; Get total size
     03     57    91 0661            CMPB    R7,#3                   ; Is it 3 null (counted) strings
            13    13 0664            BEQL    60$                     ; If so use access info in OBI
            00    90 0666            MOVB    #NMA$C_ACES_NONE,-      ; Disallow proxy access
        004A'CF      0668                      INT_B_PRX             ; Store it
     75 8F   57    91 066B            CMPB    R7,#NET$C_MAXACCFLD*3   ; Too long ?
            13    1B 066F            BLEQU   70$                     ; If LEQU then move the strings
            2B    3C 0671            MOVZWL  #NET$C_DR_IMLONG,-      ; Indicate network failure type
        0008'CF      0673                      NET_L_REASON          ;
          013A    31 0676  55$:      BRW     100$                    ; Continue
                     0679                                            ;
                     0679  60$:      $GETFLD obi,s,iac               ; Get inbound access control
                     0684  70$:      ;
                     0684            ;
                     0684            ;    Enter the flags word followed by the access control strings
                     0684            ;
                     0684            ;
 53     0040'CF   D0 0684            MOVL    NET_Q_ACC+4,R3          ; Get pointer to access control buffer
            83    B4 0689            CLRW    (R3)+                   ; Clear the flags word
                     068B            $DISPATCH  TYPE=B,INT_B_PRX -   ; Don't set flag if proxy disallowed
                     068B            <-
                     068B              <NMA$C_ACES_OUTG, 80$>-
                     068B              <NMA$C_ACES_NONE, 80$>-
                     068B            >
                     0697            $DISPATCH  TYPE=B,OBI_B_PRX -   ; Don't set flag if proxy disallowed
                     0697            <-
                     0697              <NMA$C_ACES_OUTG, 80$>-
                     0697              <NMA$C_ACES_NONE, 80$>-
                     0697            >
      FE A3    01    A8 06A3            BISW    #1,-2(R3)               ; Say "proxy login allowed"
 63   68    57    28 06A7  80$:      MOVC3   R7,(R8),(R3)            ; Move access control strings,
                     06AB                                            ; even if it's null
 003C'CF    53    C0 06AB            ADDL    R3,NET_Q_ACC            ; Complete string size calc.
          024B    30 06B0            BSBW    UP_CASE                 ; Up-case all pertinent strings
                     06B3            ;
                     06B3            ;    Attempt to find an available server process which is waiting
                     06B3            ;    for a connect which matches it's context.
                     06B3            ;
 5B     0000'CF   D0 06B3            MOVL    NET$GL_CNR_SPI,R11      ; Get root of SPI database
            5A    D4 06B8            CLRL    R10                     ; Start at beginning of list
            58    D4 06BA  81$:      CLRL    R8                      ; Search key is zero
                     06BC            $SEARCH neq,spi,l,irp           ; Find an SPI with an IRP NE 0
         03 50    E8 06CA            BLBS    R0,82$                  ; Br if found, check process
          0082    31 06CD            BRW     89$                     ; Else, create process
         34 A6    D5 06D0  82$:      TSTL    XWB$L_PID(R6)           ; Is this connect "tagged" for a
            14    13 06D3            BEQL    83$                     ; specific process?
                     06D5            $GETFLD spi,l,pid               ; If so, get PID of this server
         D7 50    E9 06E0            BLBC    R0,81$                  ; (if not present, error, skip entry)
      34 A6    58    D1 06E3            CMPL    R8,XWB$L_PID(R6)        ; Is this server the intended process?
            D1    12 06E7            BNEQ    81$                     ; If not, then continue searching
                     06E9  83$:      ;
                     06E9            ;    Always check the access control, even for processes started
                     06E9            ;    with proxy requested.  This way, if different default access
                     06E9            ;    control is used (each object can specify a unique account,
                              06E9        ;   including NONE), the wrong process isn't matched.
                              06E9        ;
                              06E9        $GETFLD spi,s,acs                       ; Get ACS for server process
                 C3 50    E9  06F4        BLBC    R0,81$                          ; (if not present, error, skip entry)
          50  003C'CF     7D  06F7        MOVQ    NET_Q_ACC,R0                    ; Get access string for new connect
  61   50  00   68   57   2D  06FC        CMPC5   R7,(R8),#0,R0,(R1)              ; Does it match?
                     B6   12  0702        BNEQ    81$                             ; If no match, keep searching
                              0704        ;
                              0704        ;   Make sure the process's "proxy request" flag matches.
                              0704        ;
                              0704        $GETFLD spi,v,prl                       ; Get proxy login flag
                 A8 50    E9  070F        BLBC    R0,81$                          ; (if not present, error, skip entry)
  58   0040'DF   01  00   ED  0712        CMPZV   #0,#1,@NET_Q_ACC+4,R8           ; Does proxy login flag match?
                     9F   12  0719        BNEQ    81$                             ; If not, try to find another server
                              071B        ;
                              071B        ;   For logical links which request proxy access, require
                              071B        ;   that the requesting node and username match as well.
                              071B        ;
                 2F 58    E9  071B        BLBC    R8,87$                          ; If proxy requested,
                              071E        $GETFLD spi,l,rna                       ; Get remote node address for server
                 8E 50    E9  0729        BLBC    R0,81$                          ; (if not present, error, skip entry)
            58   3A A6    B1  072C        CMPW    XWB$W_REMNOD(R6),R8             ; Is it the same node as the connect?
                     88   12  0730        BNEQ    81$                             ; If not, try to find another server
                              0732        $GETFLD spi,s,rid                       ; Get remote user ID for server
                 0F 50    E9  073D        BLBC    R0,88$                          ; (if not present, error, skip entry)
              50 6F A6    9A  0740        MOVZBL  XWB$B_RID(R6),R0                ; Get length of RID for new connect
70 A6   50   00   68 57   2D  0744        CMPC5   R7,(R8),#0,R0,XWB$T_RID(R6) ; Does it match?
                     02   12  074B        BNEQ    88$                             ; If no match, then skip it
                     65   11  074D  87$:  BRB     SEND_TO_SERVER                  ; Server ok, send it the connect
                              074F
                   FF68   31  074F  88$:  BRW     81$                             ; (Branch helper to top of loop)
                              0752  89$:  ;
                              0752        ;
                              0752        ;   Create the user process
                              0752        ;
                              0752        $CREPRC_S         -                     ; create a process
                              0752            INPUT=  NET_Q_PROC,-                ; Network NETSERVER.COM filename
                              0752            OUTPUT= NET_Q_ACC,-                 ; Access control strings
                              0752            ERROR=  NET_Q_NCB,-                 ; 1st NCB (solely for LOGIN proxy use)
                              0752            PRCNAM= NET_Q_PRC,-                 ; Process name
                              0752            IMAGE=  NET_Q_IMAGE,-               ; Image (LOGINOUT) to run first
                              0752            PIDADR= NET_L_PID,-                 ; Place to store process id
                              0752            BASPRI= G^SYS$GB_DEFPRI,-           ; Priority
                              0752            UIC=    #<^O10@16+^O40>,-           ; UIC is [10,40]
                              0752            STSFLG= #<STS_M_NETLOG>,-           ; This is a network process
                              0752            MBXUNT= MBX_UNIT                    ; MBX for termination
                              078E                                                ; notification
                 07 50    E8  078E        BLBS    R0,90$                          ; If LBS process was created
                     01   3C  0791        MOVZWL  #NET$C_DR_RSU,-                 ; Assume because couldn't get
                0008'CF       0793                NET_L_REASON                    ; the resources
                     1B   11  0796        BRB     100$                            ; Take common exit
          50  0004'CF     D0  0798  90$:  MOVL    NET_L_PID,R0                    ; Get the EPID returned by CREPRC
                 58  50   D0  079D        MOVL    R0,R8                           ; Save EPID
              00000000'GF 16  07A0        JSB     G^EXE$EPID_TO_IPID              ; Convert to internal PID format
        0004'CF      50   D0  07A6        MOVL    R0,NET_L_PID                    ; Use internal format of PID
                     04   3C  07AB        MOVZWL  #NETUPD$_PROCRE,-               ; Say "process created"
                0000'CF       07AD                NET_L_FCT                       ;
```

```
                 07B0        ;
                 07B0        ;       The network process is created.  Now create an SPI database entry
                 07B0        ;       so we can keep track of it.
                 07B0        ;
0081        30   07B0            BSBW    CREATE_SPI                ; Create SPI database entry
                 07B3                                              ; Ignore errors if can't be inserted
            05   07B3   100$:    RSB                               ; Common exit
```

```
                              07B4  .SBTTL  SEND_TO_SERVER  - Send connect to waiting server
                              07B4  ;+
                              07B4  ;
                              07B4  ; There is a waiting server which can handle the incoming connect.  Set
                              07B4  ; it up so that the server can accept the logical link.
                              07B4  ;
                              07B4  ; Inputs:
                              07B4  ;
                              07B4  ;           R11 = SPI CNR address
                              07B4  ;           R10 = CNF for server database entry
                              07B4  ;
                              07B4  ;-
                              07B4  SEND_TO_SERVER:
           56    5A   D0      07B4          MOVL     R10,R6                          ; Save address of old CNF
              F846'   30      07B7          BSBW     NET$GETUTLBUF                   ; Get permission tc use utility buffer
              F843'   30      07BA          BSBW     CNF$INIT_UTL                    ; Initialize utility buffer
           58    56   D0      07BD          MOVL     R6,R8                           ; Pass address of old CNF
              F83D'   30      07C0          BSBW     CNF$COPY                        ; Copy old CNF to new CNF space
      57   0024'CF    7D      07C3          MOVQ     NET_Q_NCB,R7                    ; Get descriptor of NCB
                              07C8          $PUTFLD  spi,s,ncb                       ; Store it
      57   0034'CF    7D      07D3          MOVQ     NET_Q_TSK,R7                    ; Get procedure filespec
                              07D8          $PUTFLD  spi,s,sfi                       ; Store it
      57   002C'CF    7D      07E3          MOVQ     NET_Q_PRC,R7                    ; Get process name
                              07E8          $PUTFLD  spi,s,pnm                       ; Store it
              F80A'   30      07F3          BSBW     CNF$INSERT                      ; Insert new CNF (R10 = UTILBUF)
                              07F6                                                   ; and delete old CNF (R6)
                              07F6                                                   ; returns: R10 = valid CNF
                              07F6          $GETFLD  spi,l,pid                       ; Get PID of server process
   0004'CF       58   D0      0801          MOVL     R8,NET_L_PID                    ; Make it seem as if it was just created
                              0806          $GETFLD  spi,l,irp                       ; Get waiting DECLSERV IRP
              F7EC'   30      0811          BSBW     CNF$CLR_FIELD                   ; and clear it from database
           53    58   D0      0814          MOVL     R8,R3                           ; Get IRP address
       38 A3    00'   D0      0817          MOVL     S^#SS$_NORMAL,IRP$L_IOST1(R3) ; Set success into IRP
  3C A3  0004'CF      D0      081B          MOVL     NET_L_PID,IRP$L_IOST2(R3) ; Return IPID of SPI process as well
      55    1C A3     D0      0821          MOVL     IRP$L_UCB(R3),R5                ; Get UCB address
    00000000'GF       16      0825          JSB      G^COM$POST                      ; and complete the request
              F7D2'   30      082B          BSBW     NET$DEC_TRANS                   ; Account for completed transaction
   0000'CF       04   3C      082E          MOVZWL   #NETUPD$_PROCRE,NET_L_FCT ; Tell NETDRIVER that process exists
                      05      0833          RSB
```

```
                              0834  .SBTTL  CREATE_SPI        - Create SPI database entry
                              0834  ;+
                              0834  ;
                              0834  ; Subroutine to create an SPI database entry after having just created
                              0834  ; the network process.
                              0834  ;
                              0834  ; Inputs:
                              0834  ;
                              0834  ;       R6 = XWB address
                              0834  ;       Own storage
                              0834  ;
                              0834  ; Outputs:
                              0834  ;
                              0834  ;       R0 = Status code
                              0834  ;-
                              0834  CREATE_SPI:
              F7C9'   30      0834          BSBW    NET$GETUTLBUF               ; Get permission to use utility buffer
      5B   0000'CF    D0      0837          MOVL    NET$GL_CNR_SPI,R11          ; Get root of SPI database
              F7C1'   30      083C          BSBW    CNF$INIT_UTL                ; Init utility buffer as a CNF block
      58   0004'CF    D0      083F          MOVL    NET_L_PID,R8                ; Get PID of server process
                              0844          $PUTFLD spi,l,pid                   ; Store parameter into entry
58  0040'DF   01   00   EF    084F          EXTZV   #0,#1,@NET_Q_ACC+4,R8       ; Get proxy flag sent to LOGIN
                              0856          $PUTFLD spi,v,prl                   ; Store it
      57   003C'CF    7D      0861          MOVQ    NET_Q_ACC,R7                ; Get access control sent to LOGIN
                              0866          $PUTFLD spi,s,acs                   ; Store ACS string sent to LOGIN
        58    3A A6   3C      0871          MOVZWL  XWB$W_REMNOD(R6),R8         ; Get remote node address
                              0875          $PUTFLD spi,l,rna                   ; Store it
        57    6F A6   9A      0880          MOVZBL  XWB$B_RID(R6),R7            ; Make descriptor of RID
        58    70 A6   9E      0884          MOVAB   XWB$T_RID(R6),R8
                              0888          $PUTFLD spi,s,rid                   ; Store it
                 56   D4      0893          CLRL    R6                          ; No "old" CNF entry
              F768'   30      0895          BSBW    CNF$INSERT                  ; Insert into database
                      05      0898          RSB
```

```
                       0899  .SBTTL  GET_PR_NAM      - Get name of object procedure
                       0899  .SBTTL  GET_PR_ZNA      - Construct ZNA string for an object
                       0899  ;+
                       0899  ;
                       0899  ; Inputs:
                       0899  ;
                       0899  ;       R1 = Address of local task specifier
                       0899  ;
                       0899  ; Outputs:
                       0899  ;
                       0899  ;       R7/R8 = Descriptor of resultant string
                       0899  ;-
                       0899
                       0899                  .ENABL  LSB
                       0899  GET_PR_NAM:                                    ; Get procedure name
58    011C'CF     9E   0899          MOVAB   ZNABUF,R8                      ; Setup buffer pointer
          53  58  D0   089E          MOVL    R8,R3                          ; Make a copy
              0C  11   08A1          BRB     5$                             ; Continue
                       08A3  GET_PR_ZNA:
58    011C'CF     9E   08A3          MOVAB   ZNABUF,R8                      ; Point to ZNA buffer
          53  58  D0   08A8          MOVL    R8,R3                          ; Make a copy
   83     01  A1  90   08AB          MOVB    1(R1),(R3)+                    ; Enter object type
      50      81  33   08AF  5$:     CVTWB   (R1)+,R0                       ; Get format type, skip over object type
              07  12   08B2          BNEQ    20$                            ; If NEQ then not numbered object
           FF A1  95   08B4  10$:    TSTB    -1(R1)                         ; Is object type zero ?
              1D  13   08B7          BEQL    40$                            ; If EQL then error
              28  11   08B9          BRB     60$                            ; Else we're done
           FF A1  95   08BB  20$:    TSTB    -1(R1)                         ; Is object type zero ?
              16  12   08BE          BNEQ    40$                            ; If NEQ then error
      50      01  91   08C0          CMPB    #1,R0                          ; Format type 1 is a counted string
              07  13   08C3          BEQL    30$                            ; If EQL then go move the string
      50      02  91   08C5          CMPB    #2,R0                          ; Format type 2 is UIC + counted string
              0C  12   08C8          BNEQ    40$                            ; If NEQ then format type is unknown
              81  D5   08CA          TSTL    (R1)+                          ; Skip over UIC
      50      81  9A   08CC  30$:    MOVZBL  (R1)+,R0                       ; Get taskname string size
              05  13   08CF          BEQL    40$                            ; If EQL then illegal format
      0C      50  91   08D1          CMPB    R0,#MAX_TASKNAM                ; Is it within bounds?
              09  1B   08D4          BLEQU   50$                            ; If LEQU then legal format
              50  D4   08D6  40$:    CLRL    R0                             ; Else, indicate error
              05  3C   08D8          MOVZWL  #NET$C_DR_FMT,-                ; Setup network failure code
         0008'CF       08DA                          NET_L_REASON           ;
              0B  11   08DD          BRB     70$                            ; Take common exit
63    61      50  28   08DF  50$:    MOVC3   R0,(R1),(R3)                   ; Enter string
57    53      58  C3   08E3  60$:    SUBL3   R8,R3,R7                       ; Get string size
      50      01  D0   08E7          MOVL    #1,R0                          ; Indicate success
                  05   08EA  70$:    RSB
                       08EB
                       08EB                  .DSABL  LSB
```

```
                                08EB .SBTTL  TELL_DRV        - Call NETDRIVER
                                08EB ;++
                                08EB ;
                                08EB ; Call NETDRIVER to perform a given function.
                                08EB ;
                                08EB ; Inputs:
                                08EB ;
                                08EB ;       NET_L_R0-R5 = Arguments to NETDRIVER function
                                08EB ;--
                                08EB TELL_DRV:                                   ; Tell driver about process
          50   0000'CF    7D    08EB         MOVQ    NET_L_R0,R0                 ; Get regs for call
          52   0008'CF    7D    08F0         MOVQ    NET_L_R2,R2                 ;
          54   0010'CF    7D    08F5         MOVQ    NET_L_R4,R4                 ;
                  F703'   30    08FA         BSBW    CALL_NETDRIVER              ; Call driver
                          05    08FD         RSB
```

```
                              08FE  .SBTTL  UP_CASE - Upcase the LOGINOUT strings
                              08FE  ;+
                              08FE  ;
                              08FE  ; The NCB (up to the "/"), the access control strings, the taskname, and the
                              08FE  ; process name are up-cased in place.
                              08FE  ;
                              08FE  ; INPUTS:         none
                              08FE  ;
                              08FE  ; OUTPUTS:        none
                              08FE  ;
                              08FE  ;                 All register contents are preserved.
                              08FE  ;
                              08FE  ;-
                              08FE  UP_CASE:                                      ; Up-case strings passed to LOGINOUT
                     3F   BB  08FE          PUSHR   #^M<R0,R1,R2,R3,R4,R5>      ; Save regs
        55    0000'CF   9E  0900          MOVAB   NET$AB_UPASCNUM,R5          ; Get translation table
              54    2F   90  0905          MOVB    #^A'/',R4                   ; Setup terminator
        53    0024'CF   9E  0908          MOVAB   NET_Q_NCB,R3                ; Point to NCB descriptor
                     3C   10  090D          BSBB    UP_IT                       ; Up-case it in place
                     54   D4  090F          CLRL    R4                          ; Say "no terminator"
        53    0034'CF   9E  0911          MOVAB   NET_Q_TSK,R3                ; Point to task-name descriptor
                     33   10  0916          BSBB    UP_IT                       ; Up-case it in place
        53    002C'CF   9E  0918          MOVAB   NET_Q_PRC,R3                ; Point to process-name descriptor
                     2C   10  091D          BSBB    UP_IT                       ; Up-case it in place
        53    003C'CF   9E  091F          MOVAB   NET_Q_ACC,R3                ; Get access control descriptor
          51    04   A3   D0  0924          MOVL    4(R3),R1                    ; Get pointer to strings
              51    02   C0  0928          ADDL    #2,R1                       ; Skip over flags word
              52    81   9A  092B          MOVZBL  (R1)+,R2                    ; Get count of bytes in username
                     36   10  092E          BSBB    UP_CASE_LOOP                ; Start at end of loop
              52    81   9A  0930          MOVZBL  (R1)+,R2                    ; Get count of bytes in password
                     31   10  0933          BSBB    UP_CASE_LOOP                ; Start at end of loop
              52    81   9A  0935          MOVZBL  (R1)+,R2                    ; Get count of bytes in account name
                     2C   10  0938          BSBB    UP_CASE_LOOP                ; Start at end of loop
     7E    04  A3    63   C1  093A          ADDL3   (R3),4(R3),-(SP)            ; Get address of end of strings
              8E    51   D1  093F          CMPL    R1,(SP)+                    ; Have we gone too far?
                     03   1A  0942          BGTRU   10$                         ; If GTRU then yes
                     3F   BA  0944          POPR    #^M<R0,R1,R2,R3,R4,R5>      ; Restore regs
                     05       0946          RSB
                              0947
                              0947
                              0947  10$:    BUG_CHECK  NETNOSTATE,FATAL       ; Access control strings setup
                              094B                                            ; incorrectly
                              094B
                              094B
                              094B                  .ENABL  LSB
                              094B
              52    63   3C  094B  UP_IT:  MOVZWL  (R3),R2                     ; Get string length
          51    04   A3   D0  094E          MOVL    4(R3),R1                    ; Point to string
                     12   11  0952          BRB     UP_CASE_LOOP                ; Start at end of loop
              50    81   90  0954  20$:    MOVB    (R1)+,R0                    ; Get next character
              54    50   91  0957          CMPB    R0,R4                       ; Is it the terminator?
                     0D   13  095A          BEQL    60$                         ; If EQL yes, we're done
           50    6540   90  095C          MOVB    (R5)[R0],R0                 ; Up-case it
                     04   13  0960          BEQL    UP_CASE_LOOP                ; If EQL then not alpha-numeric
        FF A1    50   90  0962          MOVB    R0,-1(R1)                   ; Store up-cased value
                              0966
                              0966  UP_CASE_LOOP:
```

```
                  EB 52  F4  0966  1557         SOBGEQ  R2,20$                  ; Loop for each character
                         05  0969  1558 60$:    RSB                             ; Done
                             096A  1559
                             096A  1560                 .DSABL  LSB
                             096A  1561
                             096A  1562
                             096A  1563 .END
```

```
$$T1                       = 00000000                  LSB                          = 00000000
ACC                        = 0000000C                  LSB$B_R_CXBCNT               = 00000028
ACC$K_TERMLEN              = 00000054                  LSB$B_R_CXBQUO               = 00000029
ACP$C_STA_F                = 00000004                  LSB$B_SPARE                  = 0000002A
ACP$C_STA_H                = 00000005                  LSB$B_STS                    = 0000002B
ACP$C_STA_I                = 00000000                  LSB$B_X_ADJ                  = 0000000B
ACP$C_STA_N                = 00000001                  LSB$B_X_CXBACT               = 0000000D
ACP$C_STA_R                = 00000002                  LSB$B_X_CXBCNT               = 0000000F
ACP$C_STA_S                = 00000003                  LSB$B_X_CXBQUO               = 0000000E
BBUF                         0000010E R      02        LSB$B_X_PKTWND               = 0000000C
BIT...                     = 00000006                  LSB$B_X_REQ                  = 0000000A
BUG$_ACPMBFAIL               ********    X   04        LSB$L_CROSS                  = 0000002C
BUG$_NETNOSTATE              ********    X   05        LSB$L_R_CXB                  = 00000020
BUILD_NCB                    000004BC R      04        LSB$L_R_IRP                  = 0000001C
CALL_NETDRIVER               ********    X   04        LSB$L_X_CXB                  = 00000018
CNF                        = 00000024                  LSB$L_X_IRP                  = 00000014
CNF$CLR_FIELD                ********    X   04        LSB$L_X_PND                  = 00000010
CNF$COPY                     ********    X   04        LSB$M_BOM                    = 00000020
CNF$C_LENGTH               = 00000024                  LSB$M_EOM                    = 00000040
CNF$DELETE                   ********    X   05        LSB$M_LI                     = 00000001
CNF$GET_FIELD                ********    X   04        LSB$S_LSB                    = 00000030
CNF$INIT_UTL                 ********    X   04        LSB$S_SPARE                  = 00000004
CNF$INSERT                   ********    X   04        LSB$S_STS                    = 00000001
CNF$KEY_SEARCH               ********    X   05        LSB$V_BOM                    = 00000005
CNF$PURGE                    ********    X   05        LSB$V_EOM                    = 00000006
CNF$PUT_FIELD                ********    X   05        LSB$V_LI                     = 00000000
CNF$_ADVANCE               = 00000000                  LSB$V_SPARE                  = 00000001
CNF$_QUIT                  = 00000002                  LSB$W_HAA                    = 00000008
CNF$_TAKE_CURR             = 00000003                  LSB$W_HAR                    = 00000006
CNF$_TAKE_PREV             = 00000001                  LSB$W_HAX                    = 00000026
COM$POST                     ********    X   04        LSB$W_HNR                    = 00000024
CONNECT                      000001D7 R      04        LSB$W_HXS                    = 00000004
CONN_SPACE                 = 000003E8                  LSB$W_LNX                    = 00000002
CREATE_LLI                   00000000 R      04        LSB$W_LUX                    = 00000000
CREATE_SPI                   00000834 R      04        LTB$L_SLOTS                  = 00000010
DEAL_XWB                     00000090 R      05        LTB$L_SLT_NXT                = 00000000
DELPROC                      000001E2 R      04        LTB$L_XWB                    = 0000000C
DET_AB_ACC                   00000044 R      02        MAX_TASKNAM                  = 0000000C
DET_C_ACC                  = 00000005                  MBX_ACTION                     000000E6 R      04
DLE$PRC_EXIT                 ********    X   04        MBX_CHAN                       0000004C R      02
ENDBUF                       0000011C R      02        MBX_IOSB                       00000050 R      02
EXE$EPID_TO_IPID             ********    X   04        MBX_LEN                        00000052 R      02
EXIT_BUF                     00000018 R      03        MBX_MSG_LTH                  = 00000096
EXIT_CODE                    0000005C R      02        MBX_PID                        00000054 R      02
EXIT_ID                      00000058 R      02        MBX_RDCNT                      0000004E R      02
EXIT_MSG                     00000058 R      02        MBX_UNIT                       0000011A R      02
GET_PROC                     0000055A R      04        MSG$_CONNECT                 = 00000032
GET_PR_NAM                   00000899 R      04        MSG$_DELPROC                 = 00000003
GET_PR_ZNA                   000008A3 R      04        MSG$_PATHLOST                = 00000036
INT_B_PRX                    0000004A R      02        MSG$_RESET                   = 00000041
IO$_READVBLK                 ********    X   04        NCB_DATA                       0000005C R      02
IRP$L_IOST1                = 00000038                  NDC$C_LENGTH                 = 0000001C
IRP$L_IOST2                = 0000003C                  NET$AB_UPASCNUM                ********    X   04
IRP$L_UCB                  = 0000001C                  NET$ACQUIRE_NDCOU              ********    X   04
LGI$_INVPWD                  ********    X   04        NET$ALLOCATE                   ********    X   04
LLI$Z_NDC_LZ               = 00000024                  NET$ALONPAGED                  ********    X   04
LLI$Z_NDC_RT               = 00000008                  NET$BIN2ASC                    ********    X   04
```

```
NET$CONNECT_FAIL                  0000024C RG    04      NET$SET_MBX_AST                   000000C9 RG    04
NET$CREATE_MBX                    0000004C RG    04      NET$STARTUP_OBJ                   00000317 RG    04
NET$C_ACT_TIMER                 = 0000001E               NET$STARTUP_OBJ_NAM               000002FD RG    04
NET$C_DR_ACCESS                 = 00000022               NETUPD$_ABORT                   = 00000001
NET$C_DR_EXIT                   = 00000026               NETUPD$_CONNECT                 = 00000002
NET$C_DR_FMT                    = 00000005               NETUPD$_EXIT                    = 00000003
NET$C_DR_IMLONG                 = 0000002B               NETUPD$_PROCRE                  = 00000004
NET$C_DR_NOBJ                   = 00000004               NET_A_LLI                         00000020 R     02
NET$C_DR_RSU                    = 00000001               NET_A_NCB                         00000010 R     02
NET$C_DYN_WQE                     ********    X  02      NET_L_FCT                         00000000 R     02
NET$C_EFN_ASYN                  = 00000002               NET_L_LNK                         0000000C R     02
NET$C_EFN_WAIT                  = 00000001               NET_L_LPD                         00000004 R     02
NET$C_IPL                       = 00000008               NET_L_PID                         00000004 R     02
NET$C_MAXACCFLD                 = 00000027               NET_L_R0                          00000000 R     02
NET$C_MAXLINNAM                 = 0000000F               NET_L_R1                          00000004 R     02
NET$C_MAXLNK                    = 000003FF               NET_L_R2                          00000008 R     02
NET$C_MAXNODNAM                 = 00000006               NET_L_R3                          0000000C R     02
NET$C_MAXOBJNAM                 = 0000000C               NET_L_R4                          00000010 R     02
NET$C_MAX_AREAS                 = 0000003F               NET_L_R5                          00000014 R     02
NET$C_MAX_LINES                 = 00000040               NET_L_REASON                      00000008 R     02
NET$C_MAX_NCB                   = 0000006E               NET_L_UCB                         00000014 R     02
NET$C_MAX_NODES                 = 000003FF               NET_Q_ACC                         0000003C R     02
NET$C_MAX_OBJ                   = 000000FF               NET_Q_IMAGE                       00000045 R     03
NET$C_MAX_WQE                   = 00000014               NET_Q_NCB                         00000024 R     02
NET$C_MINBUFSIZ                 = 000000C0               NET_Q_NETPREFIX                   00000000 R     03
NET$C_TID_ACT                   = 00000003               NET_Q_PRC                         0000002C R     02
NET$C_TID_RUS                   = 00000001               NET_Q_PROC                        0000005B R     03
NET$C_TID_XRT                   = 00000002               NET_Q_SYSTEM                      00000032 R     03
NET$C_TRCTL_CEL                 = 00000002               NET_Q_TASKZNA                     0000000B R     03
NET$C_TRCTL_OVR                 = 00000005               NET_Q_TSK                         00000034 R     02
NET$C_UTLBUFSIZ                 = 00001000               NEW_LINK                          0000009E R     05
NET$DEALLOCATE                    ********    X  05      NFB$C_LLI_XWB                   = 08010017
NET$DECR_MCOUNT                   ********    X  05      NFB$C_NDI_NNA                   = 02020043
NET$DEC_TRANS                     ********    X  04      NFB$C_NDI_TAD                   = 02010010
NET$DELIVER_CI                    000003DC R     04      NFB$C_OBI_IAC                   = 03020043
NET$DLL_X25_CALL                  ********    X  04      NFB$C_OBI_NAM                   = 03020044
NET$DLL_X25_RESET                 ********    X  04      NFB$C_OBI_NUM                   = 03010014
NET$DRV_CANCEL                    ********    X  04      NFB$C_OBI_PID                   = 03010015
NET$FLUSH_LLI_CNT                 ********    X  05      NFB$C_OBI_PRX                   = 03010016
NET$GETUTLBUF                     ********    X  04      NFB$C_OBI_SFI                   = 03020042
NET$GL_CNR_LLI                    ********    X  05      NFB$C_OBI_UCB                   = 03010012
NET$GL_CNR_NDI                    ********    X  04      NFB$C_OBI_ZNA                   = 03020041
NET$GL_CNR_OBI                    ********    X  04      NFB$C_OP_EQL                    = 00000000
NET$GL_CNR_SPI                    ********    X  04      NFB$C_OP_NEQ                    = 00000003
NET$GL_NET_UCB                    ********    X  04      NFB$C_SPI_ACS                   = 12020041
NET$GL_PTR_VCB                    ********    X  04      NFB$C_SPI_IRP                   = 12010011
NET$GQ_MBX_NAME                   00000020 RG    03      NFB$C_SPI_NCB                   = 12020044
NET$GQ_WQE_MBX                    000000F6 RG    02      NFB$C_SPI_PID                   = 12010010
NET$KILL_MBX                      0000008B RG    04      NFB$C_SPI_PNM                   = 12020045
NET$MBX_AST                       000000EE RG    04      NFB$C_SPI_PRL                   = 12000002
NET$MBX_QIO                       00000098 RG    04      NFB$C_SPI_RID                   = 12020042
NET$M_MAXLNKMSK                 = 000003FF               NFB$C_SPI_RNA                   = 12010013
NET$PROC_XWB                      00000000 RG    05      NFB$C_SPI_SFI                   = 12020043
NET$RELEASE_NDCOU                 ********    X  05      NMA$C_ACES_BOTH                 = 00000003
NET$RESEND_SERVER                 000002CA RG    04      NMA$C_ACES_NONE                 = 00000000
NET$SCAN_FOR_ZNA                  00000287 RG    04      NMA$C_ACES_OUTG                 = 00000002
NET$SERVER_FAIL                   0000026D RG    04      NSP$C_EXT_LNK                   = 0000001E
```

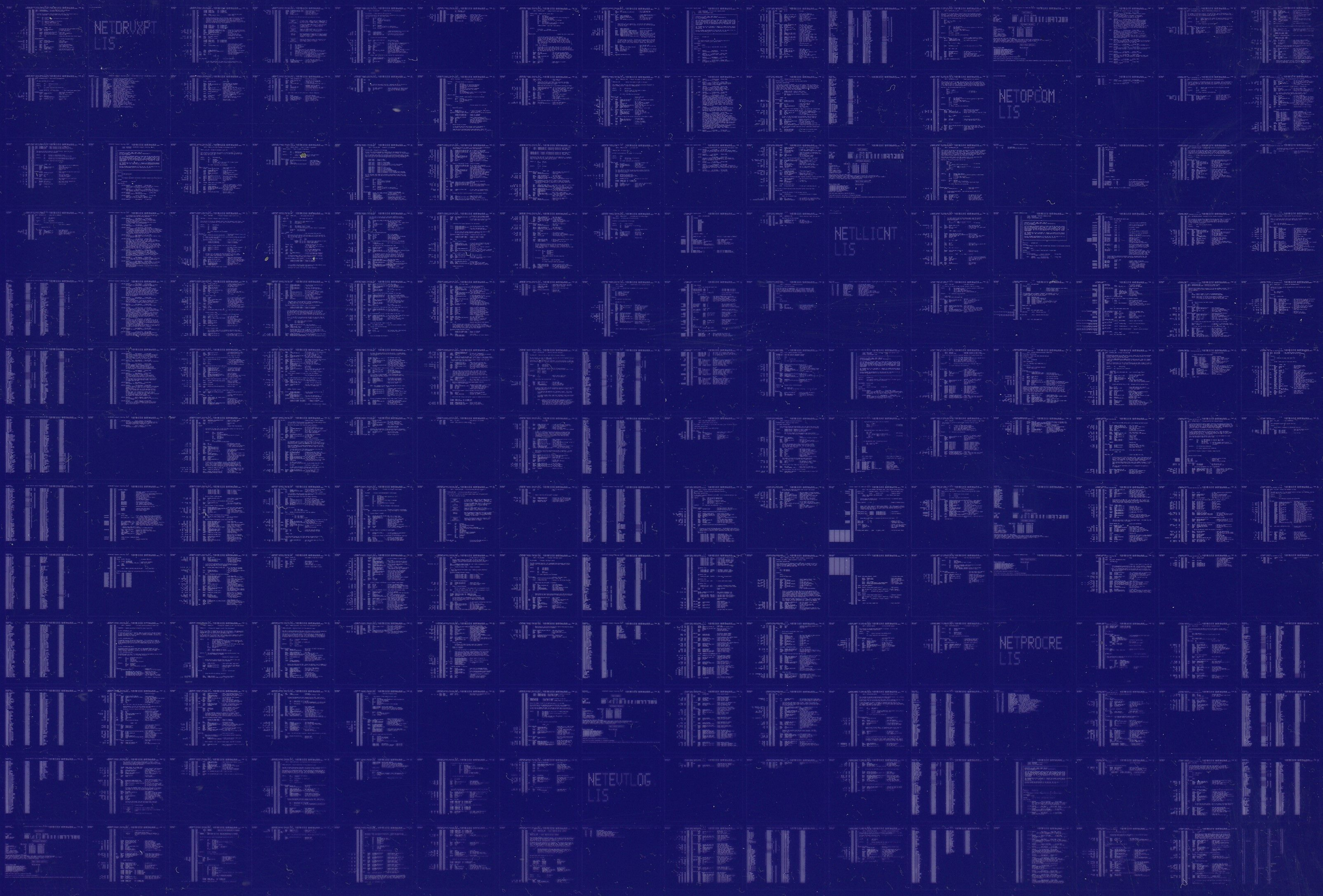
```
NSP$C_MAXHDR                     = 00000009                         XWB$B_STA                        = 0000001E
OBI_B_PRX                          00000049 R       02              XWB$B_TYPE                       = 0000000A
PR$_IPL                            ********     X   05              XWB$B_X_FLW                      = 0000006C
PRC                              = 000000C8                         XWB$B_X_FLWCNT                   = 0000006D
PTR_CON_BUF                        0000001C R       02              XWB$C_COMLNG                     = 000000A4
PTR_NCB_BUF                        00000018 R       02              XWB$C_CONLNG                     = 00000112
RCB$B_ECL_DPX                    = 00000067                         XWB$C_DATA                       = 00000010
RCB$L_PTR_LTB                    = 00000024                         XWB$C_LOGIN                      = 00000040
RCB$W_CNT_XRE                    = 0000009C                         XWB$C_LPRNAM                     = 00000014
RMS$_FNF                           ********     X   04              XWB$C_NDC_LNG                    = 00000020
SEND_TO_SERVER                     000007B4 R       04              XWB$C_NUMSTA                     = 00000008
SIZ...                           = 00000001                         XWB$C_RID                        = 00000010
SS$_ABORT                          ********     X   04              XWB$C_RPRNAM                     = 00000014
SS$_CANCEL                         ********     X   04              XWB$C_STA_CAR                    = 00000002
SS$_CONNECFAIL                     ********     X   05              XWB$C_STA_CCS                    = 00000004
SS$_NORMAL                         ********     X   04              XWB$C_STA_CIR                    = 00000003
STS$V_INHIB_MSG                  = 0000001C                         XWB$C_STA_CIS                    = 00000001
STS_M_NETLOG                     = 00000080                         XWB$C_STA_CLO                    = 00000000
STS_M_NOACNT                     = 00000008                         XWB$C_STA_DIR                    = 00000006
STS_M_NOAUTH                     = 00000040                         XWB$C_STA_DIS                    = 00000007
SYS$CREMBX                         ********    GX   04              XWB$C_STA_RUN                    = 00000005
SYS$CREPRC                         ********    GX   04              XWB$L_DEA_IRP                    = 00000104
SYS$DASSGN                         ********    GX   04              XWB$L_FPC                        = 00000020
SYS$GB_DEFPRI                      ********     X   04              XWB$L_FR3                        = 00000024
SYS$GETCHN                         ********    GX   04              XWB$L_FR4                        = 00000028
SYS$QIO                            ********    GX   04              XWB$L_ICB                        = 0000010C
TASKZNA                            00000013 R       03              XWB$L_IRP_ACC                    = 00000080
TELL_DRV                           000008EB R       04              XWB$L_LINK                       = 0000002C
TR$C_MAXHDR                      = 0000001C                         XWB$L_ORGUCB                     = 00000010
TR$C_NI_ALLEND1                  = 040000AB                         XWB$L_PID                        = 00000034
TR$C_NI_ALLEND2                  = 00000000                         XWB$L_VCB                        = 00000030
TR$C_NI_ALLROU1                  = 030000AB                         XWB$L_WLBL                       = 00000004
TR$C_NI_ALLROU2                  = 00000000                         XWB$L_WLFL                       = 00000000
TR$C_NI_PREFIX                   = 000400AA                         XWB$M_FLG_BREAK                  = 00000001
TR$C_NI_PROT                     = 00000360                         XWB$M_FLG_CLO                    = 00000200
TR$C_PRI_ECL                     = 0000001F                         XWB$M_FLG_IAVL                   = 00001000
TR$C_PRI_RTHRU                   = 0000001F                         XWB$M_FLG_SCD                    = 00000100
TSK                              = 0000012C                         XWB$M_FLG_SDACK                  = 00000008
UP_CASE                            000008FE R       04              XWB$M_FLG_SDFL                   = 00004000
UP_CASE_LOOP                       00000966 R       04              XWB$M_FLG_SDT                    = 00000080
UP_IT                              0000094B R       04              XWB$M_FLG_SIACK                  = 00000004
WQE$C_SUB_MBX                    = 00000005                         XWB$M_FLG_SIFL                   = 00002000
WQE$INSQUE                         ********     X   04              XWB$M_FLG_SLI                    = 00000010
WQE$L_PM1                        = 00000010                         XWB$M_FLG_TBPR                   = 00000800
WQE$L_PM2                        = 00000014                         XWB$M_FLG_WBP                    = 00000040
WQE_MBX_LTH                      = 00000018                         XWB$M_FLG_WBUF                   = 00000002
X25_DEV_NAME                       00000077 R       03              XWB$M_FLG_WDAT                   = 00000400
XWB                              = 00000000                         XWB$M_FLG_WHGL                   = 00000020
XWB$B_ACCESS                     = 0000000B                         XWB$M_PRO_CCA                    = 00000008
XWB$B_DATA                       = 0000005B                         XWB$M_PRO_NAR                    = 00000010
XWB$B_FIPL                       = 0000001F                         XWB$M_PRO_NFC                    = 00000001
XWB$B_LOGIN                      = 000000CC                         XWB$M_PRO_PH2                    = 00000004
XWB$B_LPRNAM                     = 000000A4                         XWB$M_PRO_SFC                    = 00000002
XWB$B_PRO                        = 0000005A                         XWB$M_STS_ASTPND                 = 00000400
XWB$B_RID                        = 0000006F                         XWB$M_STS_ASTREQ                 = 00000800
XWB$B_RPRNAM                     = 000000B8                         XWB$M_STS_CON                    = 00000010
XWB$B_SP3                        = 0000006E                         XWB$M_STS_DIS                    = 00000008
```

```
                                                          B  1
NETPROCRE                       - Process creation                     16-SEP-1984 01:27:29  VAX/VMS Macro V04-00        Page  41
Symbol table                                                            5-SEP-1984 02:21:33  [NETACP.SRC]NETPROCRE.MAR;1       (20)

XWB$M_STS_DTNAK                 = 00000100            XWB$V_PRO_PH2                = 00000002
XWB$M_STS_LINAK                 = 00000200            XWB$V_PRO_SFC                = 00000001
XWB$M_STS_NDC                   = 00001000            XWB$V_STS_ASTPND             = 0000000A
XWB$M_STS_OVF                   = 00000080            XWB$V_STS_ASTREQ             = 0000000B
XWB$M_STS_RBP                   = 00000040            XWB$V_STS_CON                = 00000004
XWB$M_STS_SOL                   = 00000004            XWB$V_STS_DIS                = 00000003
XWB$M_STS_TID                   = 00000001            XWB$V_STS_DTNAK              = 00000008
XWB$M_STS_TLI                   = 00000002            XWB$V_STS_LINAK              = 00000009
XWB$M_STS_TMO                   = 00000020            XWB$V_STS_NDC                = 0000000C
XWB$Q_FORK                      = 00000014            XWB$V_STS_OVF                = 00000007
XWB$Q_FREE_CXB                  = 00000118            XWB$V_STS_RBP                = 00000006
XWB$R_CON_BLK                   = 000000A4            XWB$V_STS_SOL                = 00000002
XWB$R_RUN_BLK                   = 000000A4            XWB$V_STS_TID                = 00000000
XWB$S_                          = 00000006            XWB$V_STS_TLI                = 00000001
XWB$S_COMLNG                    = 0000006E            XWB$V_STS_TMO                = 00000005
XWB$S_CON_BLK                   = 0000006E            XWB$W_CI_PATH                = 00000110
XWB$S_DATA                      = 00000010            XWB$W_DECAY                  = 0000004E
XWB$S_DT                        = 00000030            XWB$W_DLY_FACT               = 00000056
XWB$S_FLG                       = 00000002            XWB$W_DLY_WGHT               = 00000058
XWB$S_FORK                      = 00000008            XWB$W_ELAPSE                 = 0000004A
XWB$S_FREE_CXB                  = 00000008            XWB$W_FLG                    = 0000001C
XWB$S_LI                        = 00000030            XWB$W_LOCLNK                 = 0000003E
XWB$S_LOGIN                     = 0000003F            XWB$W_LOCSIZ                 = 00000040
XWB$S_LPRNAM                    = 00000013            XWB$W_PATH                   = 00000038
XWB$S_NDC                       = 00000020            XWB$W_PROGRESS               = 00000052
XWB$S_PRO                       = 00000001            XWB$W_REFCNT                 = 0000000C
XWB$S_RID                       = 00000010            XWB$W_REMLNK                 = 0000003C
XWB$S_RPRNAM                    = 00000013            XWB$W_REMNOD                 = 0000003A
XWB$S_RUN_BLK                   = 00000064            XWB$W_REMSIZ                 = 00000042
XWB$S_STS                       = 00000002            XWB$W_RETRAN                 = 00000054
XWB$S_XWB                       = 00000120            XWB$W_R_REASON               = 00000044
XWB$T_                          = 00000112            XWB$W_SIZE                   = 00000008
XWB$T_DATA                      = 0000005C            XWB$W_STS                    = 0000000E
XWB$T_DT                        = 000000A4            XWB$W_TIMER                  = 00000050
XWB$T_LI                        = 000000D4            XWB$W_TIM_ID                 = 00000048
XWB$T_LOGIN                     = 000000CD            XWB$W_TIM_INACT              = 0000004C
XWB$T_LPRNAM                    = 000000A5            XWB$W_X_REASON               = 00000046
XWB$T_RID                       = 00000070            XWB$Z_NDC                    = 00000084
XWB$T_RPRNAM                    = 000000B9            ZNABUF                         0000011C R      02
XWB$V_FLG_BREAK                 = 00000000            _$$_                         = 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
```

```
                                                      +-----------------+
                                                      ! Psect synopsis !
                                                      +-----------------+

PSECT name                       Allocation          PSECT No.  Attributes
----------                       ----------          ---------  ----------
.  ABS  .                        00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                            00000000  (    0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
NET_IMPURE                       00000130  (  304.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD     WRT NOVEC LONG
NET_PURE                         00000082  (  130.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD   NOWRT NOVEC LONG
NET_CODE                         0000096A  ( 2410.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC LONG
NET_LOCK_CODE                    0000012C  (  300.)  05 (  5.)  NOPIC   USR   CON   REL   GBL NOSHR   EXE   RD   NOWRT NOVEC BYTE

                                                   +------------------------+
                                                   ! Performance indicators !
                                                   +------------------------+

Phase                       Page faults    CPU Time       Elapsed Time
-----                       -----------    --------       ------------
Initialization                      34     00:00:00.05    00:00:00.31
Command processing                 129     00:00:00.98    00:00:02.95
Pass 1                             741     00:00:29.71    00:00:49.52
Symbol table sort                    0     00:00:04.21    00:00:07.91
Pass 2                             631     00:00:06.49    00:00:14.79
Symbol table output                 56     00:00:00.40    00:00:00.96
Psect synopsis output                6     00:00:00.03    00:00:00.03
Cross-reference output               0     00:00:00.00    00:00:00.00
Assembler run totals              1599     00:00:41.88    00:01:16.49

The working set limit was 900 pages.
163425 bytes (320 pages) of virtual memory were used to buffer the intermediate code.
There were 160 pages of symbol table space allocated to hold 2818 non-local and 95 local symbols.
1563 source lines were read in Pass 1, producing 32 object records in Pass 2.
65 pages of virtual memory were used to define 53 macros.

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

Macro library name                                Macros defined
------------------                                --------------
_$255$DUA28:[SHRLIB]NMALIBRY.MLB;1                          1
_$255$DUA28:[SHRLIB]EVCDEF.MLB;1                            0
_$255$DUA28:[NETACP.OBJ]NETDRV.MLB;1                        1
_$255$DUA28:[NETACP.OBJ]NET.MLB;1                          17
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                              6
_$255$DUA28:[SYSLIB]STARLET.MLB;2                          19
TOTALS (all libraries)                                     44

3081 GETS were required to define 44 macros.

There were no errors, warnings or information messages.

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

0278 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY


NETDRVXPT
LIS

NETOPCOM
LIS

NETLLICNT
LIS

NETPROCRE
LIS

NETEVTLOG
LIS

0279 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

NICNF

CNFDEF
SDL

CNFDEF
LIS

NETTRN
LIS

NICONFIG
MAP

NETTREE
LIS

CNFMAIN
LIS

CNFREQUES
LIS

SERVER
LIS

CNFINTRPT
LIS

CNFWQDEF
SDL

CNFPREFIX
REQ

CNFMSG
LIS