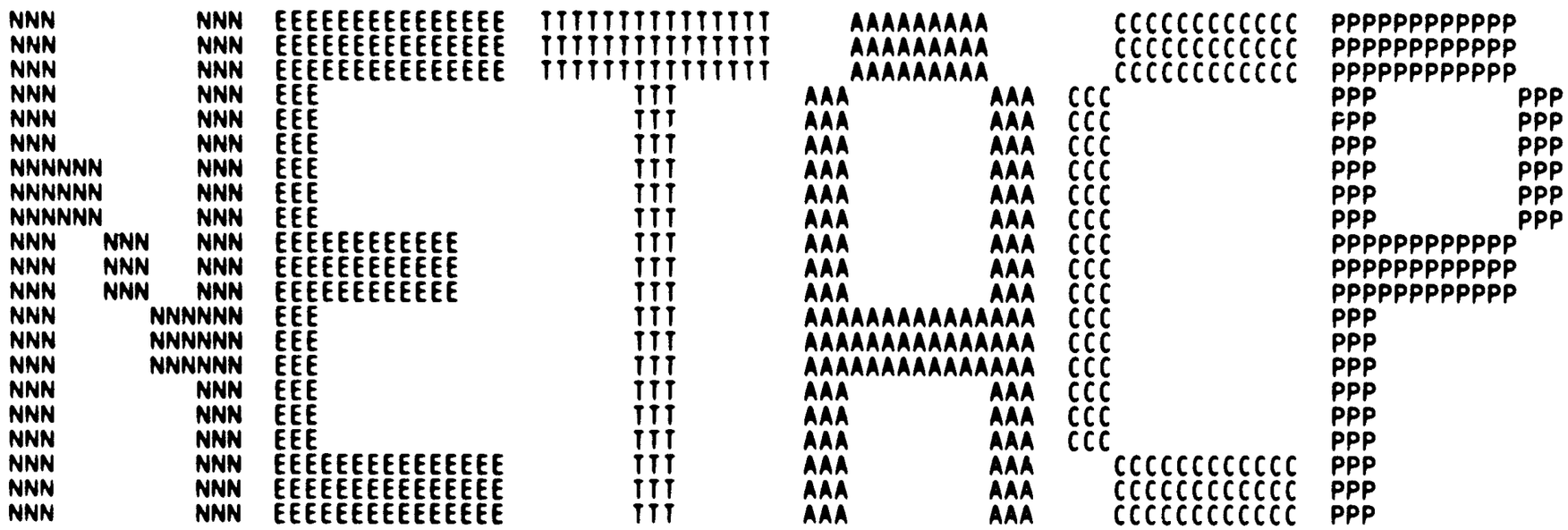
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN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

FILEID**NETPROCRE

```
NN      NN  EEEEEEEEEE  TTTTTTTTTT  PPPPPPPP    RRRRRRRR      000000      CCCCCCCC  RRRRRRRR    EEEEEEEEEE
NN      NN  EEEEEEEEEE  TTTTTTTTTT  PPPPPPPP    RRRRRRRR      000000      CCCCCCCC  RRRRRRRR    EEEEEEEEEE
NN      NN  EE              TT      PP      PP  RR      RR  00      00  CC            RR      RR  EE
NN      NN  EE              TT      PP      PP  RR      RR  00      00  CC            RR      RR  EE
NNNN    NN  EE              TT      PP      PP  RR      RR  00      00  CC            RR      RR  EE
NNNN    NN  EE              TT      PP      PP  RR      RR  00      00  CC            RR      RR  EE
NN  NN  NN  EEEEEEEE        TT      PPPPPPPP    RRRRRRRR    00      00  CC            RRRRRRRR    EEEEEEEE
NN  NN  NN  EEEEEEEE        TT      PPPPPPPP    RRRRRRRR    00      00  CC            RRRRRRRR    EEEEEEEE
NN    NNNN  EE              TT      PP          RR  RR      00      00  CC            RR  RR      EE
NN    NNNN  EE              TT      PP          RR  RR      00      00  CC            RR  RR      EE
NN      NN  EE              TT      PP          RR    RR    00      00  CC            RR    RR    EE              ....
NN      NN  EE              TT      PP          RR    RR    00      00  CC            RR    RR    EE              ....
NN      NN  EEEEEEEEEE      TT      PP          RR      RR    000000      CCCCCCCC  RR      RR  EEEEEEEEEE      ....
NN      NN  EEEEEEEEEE      TT      PP          RR      RR    000000      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
```

Table of contents

```
0000     1          .TITLE  NETPROCRE - Process creation
0000     2          .IDENT  'V04-000'
0000     3          .DEFAULT DISPLACEMENT,WORD
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY        *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29
0000    30 ;++
0000    31 ; FACILITY:     NETWORK ACP
0000    32 ;
0000    33 ; ABSTRACT:
0000    34 ;       THIS MODULE PERFORMS PROCESS CREATION FOR AN INBOUND CONNECT.
0000    35 ;
0000    36 ; ENVIRONMENT:
0000    37 ;       MODE = KERNEL
0000    38 ;
0000    39 ; AUTHOR:       SCOTT G. DAVIS, CREATION DATE: 10-AUG-77
0000    40 ;
0000    41 ; MODIFIED BY:
0000    42 ;
0000    43 ;       V03-024 ADE0039         Alan D. Eldridge        18-Jul-1984
0000    44 ;               When looking for a free XWB slot, don't allow either byte of
0000    45 ;               the local link number to be equal the character '"' since
0000    46 ;               that results in some non-intelligent NCB parsers to break.
0000    47 ;
0000    48 ;       V03-025 PRB0340         Paul Beck       18-Jul-1984  16:10
0000    49 ;               Test against LGI$_INVPWD for invalid access instead of magic number.
0000    50 ;
0000    51 ;       V03-024 ADE0038         Alan D. Eldridge        25-Jun-1984
0000    52 ;               Change SS$_NOLINKS to SS$_CONNECFAIL on problems finding
0000    53 ;               or creating logical-link resources.
0000    54 ;
0000    55 ;       V03-023 RNG0023         Rod Gamache             12-Jun-1984
0000    56 ;               Change calling conventions for calls to NODE COUNTER
0000    57 ;               BLOCK access routines.
```

```
0000     58 ;
0000     59 ;        V03-022 PRB0331         Paul Beck       1-May-1984  20:19
0000     60 ;                1.      Look for EPID instead of IPID in OBI$L_PID
0000     61 ;                2.      Fix callers of NET$DELIVER_CI to set up R0 correctly.
0000     62 ;
0000     63 ;        V03-021 ADE0001         Alan D. Eldridge        11-Apr-1984
0000     64 ;                When comparing remote link addresses in NET$PROC_XWB, ignore
0000     65 ;                an address of zero.
0000     66 ;
0000     67 ;        V03-019 PRB0317         Paul Beck       8-Mar-1984  17:36
0000     68 ;                Force created network processes to use DCL as their default
0000     69 ;                CLI, independent of the default CLI for the specified account.
0000     70 ;                Fix bug in ADE0035.
0000     71 ;
0000     72 ;        V018    ADE0035         Alan D. Eldridge        14-Feb-1984
0000     73 ;                Create LLI entry when receive notification of a new XWB.
0000     74 ;
0000     75 ;        V017    RNG0017         Rod Gamache     7-Feb-1984
0000     76 ;                Fix initialization of local storage in NET$DELIVER_CI routine.
0000     77 ;
0000     78 ;        V016    TMH0016         Tim Halvorsen   23-Jun-1983
0000     79 ;                Fix selection of waiting network processes so
0000     80 ;                that processes which were activated with different
0000     81 ;                default accounts (using default accounts on different
0000     82 ;                objects) are correctly selected.
0000     83 ;
0000     84 ;        V015    RNG0015         Rod Gamache     20-Apr-1983
0000     85 ;                Fix branch destinations out of range.
0000     86 ;
0000     87 ;        V014    TMH0014         Tim Halvorsen   03-Mar-1983
0000     88 ;                If requested object name starts with a "$", then use
0000     89 ;                a default filespec of SYS$SYSTEM (rather than SYS$LOGIN)
0000     90 ;                since objects with a "$" are reserved to DEC.
0000     91 ;                Allow STARTUP_OBJ to be called with an object name
0000     92 ;                as well as a number.
0000     93 ;                Notify new DLE module of process termination.
0000     94 ;
0000     95 ;        V013    TMH0013         Tim Halvorsen   14-Feb-1983
0000     96 ;                Remove node proxy access parameter.
0000     97 ;                Add support for EPIDs.
0000     98 ;                Return IPID of SPI database key in IOSB of DECLSERV QIO
0000     99 ;                to NETSERVER process.
0000    100 ;
0000    101 ;        V012    RNG0012         Rod Gamache     26-Jan-1983
0000    102 ;                Fix bug in NET$DELIVER_CI which doesn't check status
0000    103 ;                for success on call to memory allocation routine.
0000    104 ;
0000    105 ;        V011    TMH0011         Tim Halvorsen   28-Dec-1982
0000    106 ;                Add routine to break all links for a given process.
0000    107 ;                Do not store NCB, SFI or PNM into SPI until the link
0000    108 ;                is actually given to the process, and not when the
0000    109 ;                process is created.
0000    110 ;
0000    111 ;        V010    TMH0010         Tim Halvorsen   11-Nov-1982
0000    112 ;                Fix bug in NETSERVER startup, so that initial connects
0000    113 ;                which have been tagged for a certain process do not get
0000    114 ;                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   180 .SBTTL  DECLARATIONS
                     0000   181 ;
                     0000   182 ; INCLUDE FILES:
                     0000   183 ;
                     0000   184         $ACCDEF
                     0000   185         $AQBDEF
                     0000   186         $STSDEF
                     0000   187         $DRDEF
                     0000   188         $MSGDEF
                     0000   189         $PRVDEF
                     0000   190         $UCBDEF
                     0000   191         $IRPDEF
                     0000   192         $RCBDEF
                     0000   193         $LLIDEF
                     0000   194         $LTBDEF
                     0000   195
                     0000   196         $CNFDEF
                     0000   197         $CNRDEF
                     0000   198         $NFBDEF
                     0000   199         $NMADEF
                     0000   200         $XWBDEF
                     0000   201
                     0000   202         $WQEDEF
                     0000   203
                     0000   204         $NETSYMDEF
                     0000   205         $NETUPDDEF
                     0000   206 ;
                     0000   207 ; MACROS:
                     0000   208 ;
                     0000   209
                     0000   210 ;
                     0000   211 ; EQUATED SYMBOLS:
                     0000   212 ;
                     0000   213
00000024             0000   214         CNF = CNF$C_LENGTH              ; Short name for readability
                     0000   215
00000008             0000   216 STS_M_NOACNT    =   1@3                 ; Do not generate accounting records
00000040             0000   217 STS_M_NOAUTH    =   1@6                 ; use caller's privs/quotas at login
00000080             0000   218 STS_M_NETLOG    =   1@7                 ; bit no. for network login
00000096             0000   219 MBX_MSG_LTH =       150                 ; size of a mailbox message
0000000C             0000   220 MAX_TASKNAM =        12                 ; Max size of taskname -- the name
                     0000   221                                        ; following teh "=" in the NCB
                     0000   222
                     0000   223         ASSUME  MBX_MSG_LTH GE ACC$K_TERMLEN
```

```
                     0000   225 ;
                     0000   226 ; OWN STORAGE:
                     0000   227 ;
                 00000000   228         .PSECT  NET_IMPURE,WRT,NOEXE,LONG
                     0000   229
                     0000   230                 .ALIGN  LONG
                     0000   231 NET_L_R0:                                ;
          00000000   0000   232 NET_L_FCT:      .LONG   0               ; Function to pass to NETDRIVER
                     0004   233 NET_L_R1:                                ;
                     0004   234 NET_L_LPD:                               ; LPD of line which is starting
          00000000   0004   235 NET_L_PID:      .LONG   0               ; PID to pass to NETDRIVER
                     0008   236 NET_L_R2:                                ;
          00000000   0008   237 NET_L_REASON:   .LONG   0               ; Disconnect reason
                     000C   238 NET_L_R3:                                ;
          00000000   000C   239 NET_L_LNK:      .LONG   0               ; Link number
                     0010   240 NET_L_R4:                                ;
          00000014   0010   241 NET_A_NCB:      .BLKA   1               ; For saving address of NCB buffer
                     0014   242 NET_L_R5:                                ;
          00000000   0014   243 NET_L_UCB:      .LONG   0               ; UCB address to pass to NETDRIVER
                     0018   244
          00000000   0018   245 PTR_NCB_BUF:    .LONG   0               ; Address of NCB buffer
          00000000   001C   246 PTR_CON_BUF:    .LONG   0               ; Address of DELIVER_CI scratch buffer
                     0020   247
          00000000   0020   248 NET_A_LLI:      .LONG   0               ; Address of create LLI
 00000000 00000000   0024   249 NET_Q_NCB:      .QUAD   0               ; NCB descriptor
 00000000 00000000   002C   250 NET_Q_PRC:      .QUAD   0               ; Process descriptor
 00000000 00000000   0034   251 NET_Q_TSK:      .QUAD   0               ; Name of file to run
 00000000 00000000   003C   252 NET_Q_ACC:      .QUAD   0               ; Descriptor for 3 account
                     0044   253                                         ; strings preceded by flags word
                     0044   254
          00000005   0044   255 DET_C_ACC = 5                            ; Buffer for access control strings
              0000   0044   256 DET_AB_ACC:     .WORD   0               ; for creating detached, privileged
                00   0046   257                 .BYTE   0               ; processes.  It consists of a flags
                00   0047   258                 .BYTE   0               ; Word followed by 3 null counted
                00   0048   259                 .BYTE   0               ; strings.
                     0049   260
                00   0049   261 OBI_B_PRX:      .BYTE   0               ; OBI proxy access state
                00   004A   262 INT_B_PRX:      .BYTE   0               ; Internal proxy access state.  This is
                     004B   263                                         ; set to "none" if any conditions are
                     004B   264                                         ; detected internally (other than the
                     004B   265                                         ; values stored in the OBI or NDI)
                     004B   266                                         ; which would disallow proxy access
                     004B   267 ;
                     004B   268 ;   Fields used for termination mailbox creation, message buffering.  Be
                     004B   269 ;!  careful when modifying since some code assumes data ordering without
                     004B   270 ;!  using assumes.
                     004B   271 ;
                     004B   272                 .ALIGN  LONG
          0000004E   004C   273 M3X_CHAN:       .BLKW   1               ; Channel number of mailbox
          00000050   004E   274 MBX_RDCNT:      .BLKW   1               ; Number of reads posted to mailbox
                     0050   275 MBX_IOSB:                               ; I/O status block
          00000052   0050   276                 .BLKW   1               ;  -- status of i/o completio
          00000054   0052   277 MBX_LEN:        .BLKW   1               ;  -- length of operation here
          00000058   0054   278 MBX_PID:        .BLKL   1               ;  -- pid of process deleted
                     0058   279 EXIT_MSG:                               ; Buffer for mailbox message
          0000005A   0058   280 EXIT_ID:        .BLKW   1               ;  -- message identification
          0000005C   005A   281                 .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   333 .SBTTL  NET$PROC_XWB - Process returned XWB
                     0000   334 ;+
                     0000   335 ;
                     0000   336 ;  NETDRIVER has passed us an XWB either to be linked into the LTB and assigned
                     0000   337 ;  a local logical-link address (upon receiving an incoming connect) or to be
                     0000   338 ;  unhooked from the LTB and deallocated.
                     0000   339 ;
                     0000   340 ;  If both the XWB$W_REMLNK and XWB$W_LOCLNK fields are zero, then this request
                     0000   341 ;  comes from the NETACP code which handles the IO$_ACCESS request for Connect
                     0000   342 ;  initiates.
                     0000   343 ;
                     0000   344 ;  NETACP is responsible for the LTB maintenance and the XWB linkage in order to
                     0000   345 ;  avoid any race conditions it may have with NETDRIVER while scanning this list
                     0000   346 ;
                     0000   347 ;
                     0000   348 ;  INPUTS:       R3      XWB pointer
                     0000   349 ;
                     0000   350 ;  OUTPUTS:      All registers are clobbered
                     0000   351 ;
                     0000   352 ;-
                     0000   353         .SAVE_PSECT
                00000000   354         .PSECT  NET_LOCK_CODE,NOWRT,GBL ; Can't tolerate page faults
                     0000   355         .ENABL  LSB
                     0000   356
                     0000   357 NET$PROC_XWB::                          ; Process (deallocate) XWB
5B   0000'CF   D0    0000   358         MOVL    NET$GL_CNR_LLI,R11      ; Pick up LLI CNR
        5A     D4    0005   359         CLRL    R10                     ; No LLI CNF yet
  52    30 A3   D0    0007   360         MOVL    XWB$L_VCB(R3),R2        ; Get RCB
  55    24 A2   D0    000B   361         MOVL    RCB$L_PTR_LTB(R2),R5    ; Get LTB
        3E A3   B5    000F   362         TSTW    XWB$W_LOCLNK(R3)        ; Test local link number
          03    12    0012   363         BNEQ    2$                      ; If NEQ, XWB being returned
        0087    31    0014   364         BRW     NEW_LINK                ; If EQL, this is an incoming connect
                     0017   365 2$:       ;
                     0017   366         ;
                     0017   367         ;    Locate and Delete the LLI CNF.  Release hold on counter block
                     0017   368         ;
                     0017   369         ;
        58    53    D0    0017   370         MOVL    R3,R8                   ; Setup XWB address for search
                     001A   371         $SEARCH egl,lli,l,xwb           ; Find the corresponding LLI, if any
          2E 50     E9    0027   372         BLBC    R0,10$                  ; If LBC, not found
                     002A   373         ;
             3C     BB    002A   374         PUSHR   #^M<R2,R3,R4,R5>        ; Save regs
5E   00000064 8F     C2    002C   375         SUBL    #100,SP                 ;&Create dummy non-pageable buffer
        56    5E    D0    0033   376         MOVL    SP,R6                   ; Point to dummy buffer
        54    01    D0    0036   377         MOVL    #1,R4                   ; Say "zero XWB counters"
        55    53    D0    0039   378         MOVL    R3,R5                   ; XWB ptr for subr call
           FFC1'    30    003C   379         BSBW    NET$FLUSH_LLI_CNT       ; Flush LLI and XWB counters to node
                     003F   380                                             ; counter block
5E   00000064 8F     C0    003F   381         ADDL    #100,SP                 ;&Release stack space
             3C     BA    0046   382         POPR    #^M<R2,R3,R4,R5>        ; Restore regs
                     0048   383         ;
             58     D4    0048   384         CLRL    R8                      ; Nullify pointer
           FFB3'    30    004A   385         BSBW    CNF$PUT_FIELD           ; Erase the XWB pointer
           FFB0'    30    004D   386         BSBW    CNF$DELETE              ; Mark the entry for deletion
           FFAD'    30    0050   387         BSBW    CNF$PURGE               ; Purge the entry from the database
             5A     D4    0053   388         CLRL    R10                     ; Forget about the LLI, its gone
           FFA8'    30    0055   389         BSBW    NET$RELEASE_NDCOU       ; Release hold on counter block
```

```
                        0058   390 10$:    ;
                        0058   391         ;
                        0058   392         ;    This is an old XWB coming back to be removed and deallocated
                        0058   393         ;
                        0058   394         ;
  50    3E A3    3C     0058   395         MOVZWL  XWB$W_LOCLNK(R3),R0      ; Get link number
50    FC00 8F    AA     005C   396         BICW    #^C<NET$C_MAXLNK>,R0     ; Clear all but 'index' bits
50    10 A540    DE     0061   397         MOVAL   LTB$L_SLOTS(R5)[R0],R0   ; Get link slot
    53    60     D1     0066   398         CMPL    (R0),R3                  ; Does address match ?
          2F     12     0069   399         BNEQ    200$                     ; If NEQ, bug
                        006B   400         DSBINT  #NET$C_IPL               ; Synchronize with NETDRIVER
                        0071   401         ;
    80    01     B0     0071   402         MOVW    #1,(R0)+                 ; Set 'available' flag
  60    3E A3    B0     0074   403         MOVW    XWB$W_LOCLNK(R3),(R0)    ; Store last used link address
  51    E0 A5    9E     0078   404         MOVAB   -XWB$[_LINK -            ;
                        007C   405                 +LTB$L_XWB(R5),R1        ; Init for scan
    50    51     D0     007C   406 20$:     MOVL    R1,R0                    ; Save a copy
  51    2C A1    D0     007F   407         MOVL    XWB$L_LINK(R1),R1        ; Travel list
    53    51     D1     0083   408         CMPL    R1,R3                    ; Is this it ?
          F4     12     0086   409         BNEQ    20$                      ; If not, branch
        2C A3    D0     0088   410         MOVL    XWB$L_LINK(R3),-         ; Remove it from list
        2C A0           008B   411                 XWB$L_LINK(R0)           ;
                        008D   412         ;
                        008D   413         ENBINT                           ; Restore IPL
                        0090   414         ;
                        0090   415 DEAL_XWB:                                ; Deallocate XWB
       FF6D'    30     0090   416         BSBW    NET$DECR_MCOUNT          ; Account for link now gone
    50    53     D0     0093   417         MOVL    R3,R0                    ; Get block address for call
       FF67'    30     0096   418         BSBW    NET$DEALLOCATE           ; Deallocate the block
                05     0099   419         RSB
                        009A   420         ;
                        009A   421 200$:    BUG_CHECK NETNOSTATE,FATAL       ; Else, bad slot address
                        009E   422         ;
                        009E   423                 .DSABL  LSB
                        009E   424         ;
                        009E   425         ;
                        009E   426 NEW_LINK:                                ; Insert new XWB into LTB
                        009E   427         ;
                        009E   428         ;
                        009E   429         ;    Find a free slot in the link table (LTB).  Start from where we left
                        009E   430         ;    off last time in order to avoid using the same slots over and over
                        009E   431         ;    again.  This technique increases the interval between re-use of a
                        009E   432         ;    logical-link number -- i.e., sequence number, slot number.
                        009E   433         ;
                        009E   434         ;    Don't allow either byte of the local link number to equal '0' since
                        009E   435         ;    some non-intelligent NCB parsers mistake that for the end of the
                        009E   436         ;    NCB.
                        009E   437         ;
                        009E   438         ;    The slot vector terminates with a -1 (longword) followed by a
                        009E   439         ;    0 (longword).
                        009E   440         ;
                        009E   441         ;
    50    0000'8F    3C     009E   442         MOVZWL  #SS$_CONNECFAIL,R0       ; Assume failure
          54    65   D0     00A3   443         MOVL    LTB$L_SLT_NXT(R5),R4     ; Get first slot candidate ptr
            FD 84    E9     00A6   444 5$:      BLBC    (R4)+,5$                 ; LBC means unavailable
FFFFFFFF 8F    74    D1     00A9   445         CMPL    -(R4),#-1                ; Backup and test for end of
          22     12     00B0   446         BNEQ    10$                      ; NEQ means slot found
```

```
                54    10 A5    DE   00B2   447          MOVAL   LTB$L_SLOTS(R5),R4      ; Start from top of vector
                      FD 84    E9   00B6   448 7$:      BLBC    (R4)+,7$                ; LBC means unavailable
            FFFFFFFF 8F   74    D1   00B9   449          CMPL    -(R4),#-1               ; Backup and test for end of
                          66    13   00C0   450          BEQL    200$                    ; EQL means slot not found
             02 A4    0400 8F    A0   00C2   451 8$:      ADDW    #NET$C_MAXLNK+1,2(R4)   ; Update local link seq #
                22    02 A4    91   00C8   452          CMPB    2(R4),#^A'"'            ; Is low byte a double quote?
                          E8    13   00CC   453          BEQL    7$                      ; If EQL yes, keep scanning
                22    03 A4    91   00CE   454          CMPB    3(R4),#^A'"'            ; Is high byte a double quote ?
                          EE    13   00D2   455          BEQL    8$                      ; If EQL, bump the seq # and try again
                                     00D4   456 10$:     ;
                                     00D4   457          ;
                                     00D4   458          ;    Find position in XWB list.  At the same time, see if this is a
                                     00D4   459          ;    duplicate by matching the remote node address (the remote link
                                     00D4   460          ;    number has not been assigned yet if this is a Connect Initiate).
                                     00D4   461          ;    If its a duplicate, simply deallocate the XWB.
                                     00D4   462          ;
                                     00D4   463          ;
                50    E0 A5    9E   00D4   464          MOVAB   -XWB$L_LINK -           ;
                                     00D8   465                  +LTB$L_XWB(R5),R0       ; Init for scan
                   51    50    D0   00D8   466 30$:     MOVL    R0,R1                   ; Remember last entry
                50    2C A0    D0   00DB   467          MOVL    XWB$L_LINK(R0),R0       ; Go to next entry
                          15    13   00DF   468          BEQL    50$                     ; If EQL, at end of list
                      3A A3    B1   00E1   469          CMPW    XWB$W_REMNOD(R3),-      ; Are we going too far ?
                      3A A0         00E4   470                  XWB$W_REMNOD(R0)        ;
                          0E    1A   00E6   471          BGTRU   50$                     ; If GTRU yes, stop here
                      3C A3    B1   00E8   472          CMPW    XWB$W_REMLNK(R3),-      ; Is this it ?
                      3C A0         00EB   473                  XWB$W_REMLNK(R0)        ;
                          E9    12   00ED   474          BNEQ    30$                     ; If NEQ no, continue searching
                      3C A3    B5   00EF   475          TSTW    XWB$W_REMLNK(R3)        ; But, if =0 then no address has been
                          E4    13   00F2   476          BEQL    30$                     ; assigned; comparison was invalid
                          9A    11   00F4   477          BRB     DEAL_XWB                ; ...else duplicate connect
                                     00F6   478 50$:     ;
                                     00F6   479          ;
                                     00F6   480          ;    LTB slot and place in XWB list have been found.  Link XWB into the
                                     00F6   481          ;    LTB and setup local link number.
                                     00F6   482          ;
                                     00F6   483          ;
                                     00F6   484          DSBINT #NET$C_IPL               ; Synch with NETDRIVER
                                     00FC   485          ;
                          04    AA   00FC   486          BICW    #XWB$M_STS_SOL,-        ;
                      0E A3         00FE   487                  XWB$W_STS(R3)           ; No longer queued
            3E A3     02 A4    B0   0100   488          MOVW    2(R4),XWB$W_LOCLNK(R3)  ; Setup local link number
                84    53    D0   0105   489          MOVL    R3,(R4)+                ; Store XWB ptr in this slot
                65    54    D0   0108   490          MOVL    R4,LTB$L_SLT_NXT(R5)    ; Store scan's next starting pt.
             2C A3    50    D0   010B   491          MOVL    R0,XWB$L_LINK(R3)       ; Link tail of list to current XWB
             2C A1    53    D0   010F   492          MOVL    R3,XWB$L_LINK(R1)       ; Link XWB to head of list
                                     0113   493          ;
                                     0113   494          ENBINT                          ; Restore IPL
                                     0116   495          ;
                   FEE7'   30   0116   496          BSBW    CREATE_LLI              ; Create LLI and insert it into database
                                     0119   497                                          ; Use status as input to NET$DELIVER_CI
                      3C A3    B5   0119   498          TSTW    XWB$W_REMLNK(R3)        ; Connect Initiate ?
                          09    13   011C   499          BEQL    100$                    ; If EQL yes, return R0 to caller
                56    53    D0   011E   500          MOVL    R3,R6                   ; Else, copy XWB address
                   02B8'   30   0121   501          BSBW    NET$DELIVER_CI          ; Create LLI, and deliver connect
                                     0124   502                                          ; notification to some server
                50    01    D0   0124   503          MOVL    #1,R0                   ; Say "success"
```

```
                        05  0127   504 100$:   RSB                                 ; Done
                            0128   505
                            0128   506 200$:   BUG_CHECK  NETNOSTATE,FATAL
                            012C   507
                        00000000   508         .RESTORE_PSECT
                            0000   509
                            0000   510
                            0000   511 CREATE_LLI:                                 ; Create LLI and insert it into the list
                            0000   512         ;
                            0000   513         ;   This subroutine in required so that the  "utility buffer"  acquired
                            0000   514         ;   by the NET$GETUTLBUF co-routine will be released in a timely manner.
                            0000   515         ;
                            0000   516         ;  NOTE - the NET$ACQUIRE_NDCOU routine needs the utility buffer, so
                            0000   517         ;   we must not allocate the utility buffer until after we acquire the
                            0000   518         ;   NDC counter block.
                            0000   519         ;
                     FFFD'  30   0000   520         BSBW    NET$ACQUIRE_NDCOU       ; Inc. reference level of counter block
                   40 50    E9   0003   521         BLBC    R0,90$                  ; If LBC, problem encountered
                     FFF7'  30   0006   522         BSBW    NET$GETUTLBUF           ; Get permission to use utility buffer
                            0009   523                                             ; - the above is a co-routine call
                5B  0000'CF D0   0009   524         MOVL    NET$GL_CNR_LLI,R11      ; Pick up CNR
                     FFEF'  30   000E   525         BSBW    CNF$INIT_UTL            ; Init "utility buffer" as a CNF
                       58   53 D0   0011   526         MOVL    R3,R8                   ; Get XWB
                            0014   527         $PUTFLD lli,l,xwb                       ; ...Store it in LLI
                            001F   528         ;
                   007C 8F   BB   001F   529         PUSHR   #^M<R2,R3,R4,R5,R6>     ; Save registers
                     56  2C AA   9E   0023   530         MOVAB   CNF+LLI$Z_NDC_RT(R10),R6; Point to 'running total' counters
        66  1C  00   6E  00   2C   0027   531         MOVC5   #0,(SP),#0,#NDC$C_LENGTH,(R6) ; Zero the counters
                     56  48 AA   9E   002D   532         MOVAB   CNF+LLI$Z_NDC_LZ(R10),R6; Point to 'last zeroed' counters
        66  1C  00   6E  00   2C   0031   533         MOVC5   #0,(SP),#0,#NDC$C_LENGTH,(R6) ; Zero the counters
                       56   D4   0037   534         CLRL    R6                      ; No "old" CNF
                     FFC4'  30   0039   535         BSBW    CNF$INSERT              ; Try to put block into list
                   007C 8F   BA   003C   536         POPR    #^M<R2,R3,R4,R5,R6>     ; Restore registers
                            0040   537         ;
                     08 50  E8   0040   538         BLBS    R0,100$                 ; If LBS, okay
                     FFBA'  30   0043   539         BSBW    NET$RELEASE_NDCOU       ; Else, dec. reference to counter block
                50  0000'8F  3C   0046   540 90$:    MOVZWL  #SS$_CONNECFAIL,R0      ; Return general purpose error status
                            05   004B   541 100$:   RSB                             ; Release utility buffer, return status
                            004C   542
```

```
                     004C    544 .SBTTL  NET$CREATE_MBX  - Create ACP mailbox
                     004C    545 .SBTTL  NET$KILL_MBX    - Delete ACP mailbox
                     004C    546 .SBTTL  NET$MBX_QIO     - Issue mailbox read
                     004C    547 ;++
                     004C    548 ;
                     004C    549 ;
                     004C    550 ; *** TBS ***
                     004C    551 ;
                     004C    552 ;--
                     004C    553 NET$CREATE_MBX::
004E'CF    B4        004C    554         CLRW    MBX_RDCNT               ; Init oustanding mailbox read count
                     0050    555         $CREMBX_S -                     ; Create mailbox
                     0050    556             CHAN   = MBX_CHAN,-
                     0050    557             MAXMSG = #MBX_MSG_LTH,-
                     0050    558             BUFQUO = #<MBX_MSG_LTH*16>,-
                     0050    559             LOGNAM = NET$GQ_MBX_NAME,-  ; mailbox's logical name
                     0050    560             PROMSK = #0
  16 50    E9        0071    561         BLBC    R0,10$                  ; Br if error
                     0074    562         $GETCHN_S -                     ; Get mailbox unit number
                     0074    563             CHAN   = MBX_CHAN,-
                     0074    564             PRIBUF = EXIT_BUF
           05        008A    565 10$:    RSB                             ; Return status in R0
                     008B    566
                     008B    567
                     008B    568
                     008B    569 NET$KILL_MBX::                          ; Delete channel to mailbox
                     008B    570         $DASSGN_S  CHAN = MBX_CHAN      ; do it
           05        0097    571         RSB
                     0098    572
                     0098    573
                     0098    574 NET$MBX_QIO::                                   ; Post read to mailbox
                     0098    575 ;
                     0098    576 ;
                     0098    577 ;  This routine puts a read out on the mailbox for process termination and
                     0098    578 ;  inbound connect notifications.
                     0098    579 ;
                     0098    580 ;
                     0098    581         $QIO_S  CHAN    = MBX_CHAN,-
                     0098    582                 FUNC    = S^#IO$_READVBLK,-
                     0098    583                 EFN     = #NET$C_EFN_ASYN,-
                     0098    584                 ASTADR  = NET$SET_MBX_AST,-
                     0098    585                 IOSB    = MBX_IOSB,-
                     0098    586                 P1      = EXIT_MSG,-
                     0098    587                 P2      = #MBX_MSG_LTH
  04 50    E8        00C1    588         BLBS    R0,10$                          ; Br unless error
                     00C4    589         BUG_CHECK       ACPMBFAIL,FATAL         ;!arrgh
           05        00C8    590 10$:    RSB                                     ; return
```

```
                                00C9   592 .SBTTL  NET$SET_MBX_AST - Process mailbox AST
                                00C9   593 ;++
                                00C9   594 ;
                                00C9   595 ;
                                00C9   596 ;--
                                00C9   597 NET$SET_MBX_AST::
                         003C   00C9   598         .WORD   ^M<R2,R3,R4,R5>
                                00CB   599
 50    00F6'CF    9E           00CB   600         MOVAB   NET$GQ_WQE_MBX,R0       ; Get base of mailbox WQE
       14 A0      D5           00D0   601         TSTL    WQE$L_PM2(R0)           ; Is it active ?
          0D      12           00D3   602         BNEQ    10$                     ; If NEQ then active, there's a bug
10 A0    04 AC    D0           00D5   603         MOVL    4(AP),WQE$L_PM1(R0)     ; Get the AST parameter
   14 A0    01    CE           00DA   604         MNEGL   #1,WQE$L_PM2(R0)        ; Mark WQE busy
         FF1F'    30           00DE   605         BSBW    WQE$INSQUE              ; Queue the WQE
                  04           00E1   606         RET                             ; Done
                               00E2   607
                               00E2   608 10$:     BUG_CHECK  NETNOSTATE,FATAL     ; Signal the bug
                               00E6   609
                               00E6   610 MBX_ACTION:                          ; Enter upon WQE dispatch
       14 A5      D4           00E6   611         CLRL    WQE$L_PM2(R5)           ; Mark WQE idle
  EE'AF    00    FB           00E9   612         CALLS   #0,B^NET$MBX_AST        ; Call the mailbox processor
                  05           00ED   613         RSB
                               00EE   614 ;+
                               00EE   615 ;
                               00EE   616 ; NET$MBX_AST - THIS ROUTINE SERVICES PROCESS TERMINATIONS
                               00EE   617 ;              AND INBOUND CONNECT NOTIFICATIONS
                               00EE   618 ;
                               00EE   619 ;-
                         0000  00EE   620 NET$MBX_AST::   .WORD   0               ; Entry point
    00'  0050'CF    B1         00F0   621         CMPW    MBX_IOSB,S^#SS$_ABORT   ; Was the i/o cancelled?
            0E      13         00F5   622         BLQL    5$                      ; If so, assume mailbox going away
0000'8F  0050'CF    B1         00F7   623         CMPW    MBX_IOSB,#SS$_CANCEL    ; Try this code, too
            05      13         00FE   624         BEQL    5$                      ; If NEQ proceed
            04      10         0100   625         BSBB    10$                     ; Dispatch
          FF93      30         0102   626         BSBW    NET$MBX_QIO             ; Put out another read
                    04         0105   627 5$:     RET                             ; Done
                               0106   628         ;
                               0106   629         ;   Dispatch
                               0106   630         ;
    50    0054'CF    D0         0106   631 10$:     MOVL    MBX_PID,R0          ; Get EPID returned by MBX driver
     00000000'GF    16         010B   632         JSB     G^EXE$EPID_TO_IPID      ; Convert to internal PID
    0054'CF    50    D0         0111   633         MOVL    R0,MBX_PID              ; Use the IPID for later processing
    56    0000'CF    D0         0116   634         MOVL    NET$GL_NET_UCB,R6       ; Point to our NET channel's UCB
    5B    0058'CF    9E         011B   635         MOVAB   EXIT_IO,R11             ; Get address of mbx message
          56    8B   B0         0120   636         MOVW    (R11)+,R6               ; Get message type
          59    8B   B0         0123   637         MOVW    (R11)+,R9               ; Get unit number
          5A    8B   9A         0126   638         MOVZBL  (R11)+,R10              ; Get device name count value
    00    6B    5A   2D         0129   639         CMPC5   R10,(R11),#0,-          ; X.25 mailbox message?
007B'DF   0077'CF              012D   640                 X2$_DEV_NAME,@X25_DEV_NAME+4
                    70   13     0133   641         BEQL    20$                     ; Branch if so
          5B    5A   C0         0135   642         ADDL    R10,R11                 ; Get pointer to mbx "data"
                               0138   643         $DISPATCH  TYPE=W,R6,-          ; Dispatch on mailbox msg type
                               0138   644         <-
                               0138   645           <MSG$_DELPROC,  DELPROC>,-      ; Process termination
                               0138   646           <MSG$_CONNECT,  CONNECT>,-      ; Inbound connect
                               0138   647           <MSG$_PATHLOST, NET$DRV_CANCEL>,-; I/O channel cancelled
                               0138   648         >
```

```
                     05  01A4   649          RSB                                 ; Ignore the message
                         01A5   650
                         01A5   651 ;
                         01A5   652 ;   Dispatch on X.25 mailbox message
                         01A5   653 ;
          5B  006C'CF    9E   01A5   654 20$:     MOVAB   EXIT_ID+20,R11          ; Point to "data"
          5A  0052'CF    3C   01AA   655          MOVZWL  MBX_LEN,R10             ; Get length of mailbox message
              5A   14    C2   01AF   656          SUBL    #20,R10                 ; Subtract out overhead
                         01B2   657          $DISPATCH  TYPE=W,R6,-              ; Dispatch on mailbox msg type
                         01B2   658          <-
                         01B2   659            <MSG$_CONNECT,  NET$DLL_X25_CALL>,-   ; Incoming X.25 call
                         01B2   660            <MSG$_RESET,    NET$DLL_X25_RESET>,-  ; X.25 circuit reset
                         01B2   661          >
                     05  01D6   662          RSB                                 ; Ignore the message
```

```
                                01D7   664 ;
                                01D7   665 ;  Connect initiate message received
                                01D7   666 ;
                                01D7   667 ;
                                01D7   668 ;       The mailbox data consists of the address of the NETDRIVER update
                                01D7   669 ;       routine and the address of the XWB.
                                01D7   670 ;
                                01D7   671 CONNECT:
                56   04 AB     D0   01D7   672          MOVL    4(R11),R6               ; Get XWB address
                   50    01     D0   01DB   673          MOVL    #1,R0                   ; "Success" flag to NET$DELIVER_CI
                      01FB     30   01DE   674          BSBW    NET$DELIVER_CI          ; Deliver the inbound connect to a user
                               05   01E1   675          RSB
                                01E2   676
                                01E2   677 ;
                                01E2   678 ;  Handle network process termination
                                01E2   679 ;
                                01E2   680 DELPROC:
                                01E2   681          ;
                                01E2   682          ;  Notify netdriver of process exit
                                01E2   683          ;
                   52    04     D0   01E2   684          MOVL    #NET$C_DR_NOBJ,R2               ; Assume can't find .com file
                                01E5   685          CLRBIT  #STS$V_INHIB_MSG,EXIT_CODE      ; Ignore INHIB_MSG flag
      005C'CF   00000000'8F     D1   01EB   686          CMPL    #RMS$_FNF,EXIT_CODE             ; file-not-found?
                        11     13   01F4   687          BEQL    35$                             ; if EQL yes
                   52    26     D0   01F6   688          MOVL    #NET$C_DR_EXIT,R2               ; assume just an exit
      005C'CF   00000000'8F     D1   01F9   689          CMPL    #LGI$_INVPWD,EXIT_CODE          ; was it an access problem?
                        03     12   0202   690          BNEQ    35$                             ; if NEQ, no
                   52    22     D0   0204   691          MOVL    #NET$C_DR_ACCESS,R2             ; say so
                51   0054'CF   D0   0207   692 35$:     MOVL    MBX_PID,R1                      ; PID of exiting process
                      005E     30   020C   693          BSBW    NET$SERVER_FAIL                 ; Notify NETSERVER server gone
                               020F   694          ;
                               020F   695          ;   Remove the process from the SPI database
                               020F   696          ;
          5B   0000'CF     D0   020F   697          MOVL    NET$GL_CNR_SPI,R11              ; Get root of SPI database
                     5A    D4   0214   698          CLRL    R10                             ; Start at beginning of list
          58   0054'CF     D0   0216   699          MOVL    MBX_PID,R8                      ; Get process PID
                               021B   700          $SEARCH egl,spi,l,pid                   ; Find database entry
                18 50     E9   0228   701          BLBC    R0,50$                          ; Branch if not found
                               022B   702          $GETFLD spi,l,irp                       ; Waiting DECLSERV IRP?
                04 50     E9   0236   703          BLBC    R0,40$                          ; Branch if no IRP waiting
                               0239   704          BUG_CHECK NETNOSTATE,FATAL              ; Should never have IRP
                               023D   705                                                  ; waiting, if process deleted
                 FDC0'     30   023D   706 40$:     BSBW    CNF$DELETE                      ; Mark the CNF entry deleted
                 FDBD'     30   0240   707          BSBW    CNF$PURGE                       ; Delete all marked CNFs
                               0243   708 50$:     ;
                               0243   709          ;   Notify DLE module of process termination.
                               0243   710          ;
          58   0054'CF     D0   0243   711          MOVL    MBX_PID,R8                      ; Setup the PID
                 FDB5'     30   0248   712          BSBW    DLE$PRC_EXIT                    ; Inform DLE of process exit
                               024B   713          RSB
```

```
                        024C   715 .SBTTL  NET$CONNECT_FAIL - Notify NETDRIVER of failed link
                        024C   716 ;+
                        024C   717 ;
                        024C   718 ; An attempt to confirm a logical link has failed.  Notify NETDRIVER so that
                        024C   719 ; it can verify the user's access to the link and then notify the remote end of
                        024C   720 ; the link that the link is being broken and why.
                        024C   721 ;
                        024C   722 ;
                        024C   723 ; INPUTS:       R3      Local logical link number (0 implies connect initiate)
                        024C   724 ;               R2      Reason code to be sent in the disconnect message
                        024C   725 ;               R1      User's PID
                        024C   726 ;
                        024C   727 ; OUTPUTS:      R5-R0   Clobbered
                        024C   728 ;
                        024C   729 ;               All other registers are preserved
                        024C   730 ;
                        024C   731 ;-
                        024C   732 NET$CONNECT_FAIL::                           ; A connect attempt has failed
0014'CF   0000'CF   D0  024C   733         MOVL    NET$GL_NET_UCB,NET_L_UCB     ; Use the ACP's UCB
     000C'CF    53   3C  0253   734         MOVZWL  R3,NET_L_LNK                 ; Specify link number
                12   13  0258   735         BEQL    10$                          ; If EQL then connect initiate
     0008'CF    52   D0  025A   736         MOVL    R2,NET_L_REASON              ; Specify disconnect reason
     0004'CF    51   D0  025F   737         MOVL    R1,NET_L_PID                 ; Specify user's PID
     0000'CF    01   D0  0264   738         MOVL    #NETUPD$_ABORT,NET_L_FCT     ; Specify "link terminated"
              067F   30  0269   739         BSBW    TELL_DRV                     ; Notify NETDRIVER
                     05  026C   740 10$:     RSB
```

```
                     026D   742 .SBTTL  NET$SERVER_FAIL - Notify NETDRIVER of terminated server
                     026D   743 ;+
                     026D   744 ;
                     026D   745 ; A server process (or NETSERVER session) has terminated.  Notify NETDRIVER so
                     026D   746 ; that it can break all links that might still be pending for that process.
                     026D   747 ; This handles the case where the process was unable to confirm the link due
                     026D   748 ; to some error.
                     026D   749 ;
                     026D   750 ;
                     026D   751 ; INPUTS:        R2      Reason code to be sent in the disconnect message
                     026D   752 ;                R1      User's PID
                     026D   753 ;
                     026D   754 ; OUTPUTS:       R5-R0   Clobbered
                     026D   755 ;
                     026D   756 ;                All other registers are preserved
                     026D   757 ;
                     026D   758 ;-
                     026D   759 NET$SERVER_FAIL::                                    ; A server has terminated
0014'CF   0000'CF    D0   026D   760         MOVL    NET$GL_NET_UCB,NET_L_UCB        ; Use the ACP's UCB
     0008'CF    52    D0   0274   761         MOVL    R2,NET_L_REASON                 ; Specify disconnect reason
     0004'CF    51    D0   0279   762         MOVL    R1,NET_L_PID                    ; Specify user's PID
     0000'CF    03    D0   027E   763         MOVL    #NETUPD$_EXIT,NET_L_FCT         ; Specity "process exit"
              0665    30   0283   764         BSBW    TELL_DRV                        ; Notify NETDRIVER
                      05   0286   765         RSB
```

```
                                0287    767 .SBTTL  NET$SCAN_FOR_ZNA - Send pending connects to declared object
                                0287    768 ;+
                                0287    769 ;
                                0287    770 ; This routine is called when a task name or object is declared by a user.
                                0287    771 ; The function is to scan the XWB list for links in the Connect Initiate
                                0287    772 ; state which are intended for the object with the specified ZNA and to build
                                0287    773 ; a NCB which is given to NETDRIVER to be put in the declared name's mailbox.
                                0287    774 ;
                                0287    775 ; INPUTS:       R7,R8 = Descriptor of object ZNA being declared
                                0287    776 ;
                                0287    777 ; OUTPUTS:      None
                                0287    778 ;
                                0287    779 ;-
                                0287    780 NET$SCAN_FOR_ZNA::                          ; Find unclaimed XWBs which match ZNA
                    59    57    7D    0287    781         MOVQ    R7,R9               ; Make copy of ZNA descriptor
                56  0000'CF    D0    028A    782         MOVL    NET$GL_PTR_VCB,R6   ; Get RCB pointer
                  56    24 A6   D0    028F    783         MOVL    RCB$L_PTR_LTB(R6),R6 ; Get LTB pointer
                  55    14 A6   9E    0293    784         MOVAB   LTB$L_SLOTS+4(R6),R5 ; Point to first XWB (skip slot 0)
                        FD 85   E8    0297    785 10$:    BLBS    (R5)+,10$           ; If LBS then pointer is invalid
                  56    FC A5   D0    029A    786         MOVL    -4(R5),R6           ; Get the XWB address
                          29    13    029E    787         BEQL    30$                 ; If EQL then done
                          03    91    02A0    788         CMPB    #XWB$C_STA_CIR,-    ; In connect initiate state?
                       1E A6          02A2    789                 XWB$B_STA(R6)       ;
                          F1    12    02A4    790         BNEQ    10$                 ; If NEQ then keep looking
                51  00A5 C6    9E    02A6    791         MOVAB   XWB$T_LPRNAM(R6),R1 ; Setup for subroutine call
                    0660 8F    BB    02AB    792         PUSHR   #^M<R5,R6,R9,R10>   ; Save important regs
                       05F1    30    02AF    793         BSBW    GET_PR_ZNA          ; Get ZNA string from LRPNAM
                        0E 50  E9    02B2    794         BLBC    R0,20$              ; If LBC then field is not valid
        6A  59  00  68    57    2D    02B5    795         CMPC5   R7,(R8),#0,R9,(R10) ; Are they the same?
                          06    12    02BB    796         BNEQ    20$                 ; If NEQ keep looking
                    50    01    D0    02BD    797         MOVL    #1,R0               ; "Success" flag to NET$DELIVER_CI
                       0119    30    02C0    798         BSBW    NET$DELIVER_CI      ; Build NCB, pass to user in mailbox
                    0660 8F    BA    02C3    799 20$:    POPR    #^M<R5,R6,R9,R10>   ; Restore regs
                          CE    11    02C7    800         BRB     10$                 ; Keep looking
                                05    02C9    801 30$:    RSB                         ; Done
```

```
                     02CA   803 .SBTTL  NET$RESEND_SERVER - Re-send initial connect to server
                     02CA   804 ;+
                     02CA   805 ;
                     02CA   806 ; This routine is called when a server process declares that it is waiting
                     02CA   807 ; for an incoming connect.  The XWB list is scanned for links in the CI
                     02CA   808 ; state looking to see if the initial connect which started the process
                     02CA   809 ; is still pending.  If so, then re-send the NCB to the server process
                     02CA   810 ; so that it will be executed.
                     02CA   811 ;
                     02CA   812 ; INPUTS:       R8 = PID of server process
                     02CA   813 ;
                     02CA   814 ; OJTPUTS:      None
                     02CA   815 ;
                     02CA   816 ;-
                     02CA   817 NET$RESEND_SERVER::                          ; Find unclaimed XWBs for server process
56   0000'CF   D0    02CA   818          MOVL    NET$GL_PTR_VCB,R6          ; Get RCB pointer
  56   24 A6   D0    02CF   819          MOVL    RCB$L_PTR_LTB(R6),R6       ; Get LTB pointer
  55   14 A6   9E    02D3   820          MOVAB   LTB$L_SLOTS+4(R6),R5       ; Point to first XWB (skip slot 0)
       FD 85   E8    02D7   821 10$:     BLBS    (R5)+,10$                  ; If LBS then pointe  is invalid
  56   FC A5   D0    02DA   822          MOVL    -4(R5),R6                  ; Get the XWB addres
          1C   13    02DE   823          BEQL    30$                        ; If EQL then done
          03   91    02E0   824          CMPB    #XWB$C_STA_CIR,-           ; In connect initiate state?
       1E A6        02E2   825                  XWB$B_STA(R6)              ;
          F1   12    02E4   826          BNEQ    10$                        ; If NEQ then keep looking
34 A6     58   D1    02E6   827          CMPL    R8,XWB$L_PID(R6)           ; Intended for this process?
          EB   12    02EA   828          BNEQ    10$                        ; If NEQ keep looking
     0160 8F   BB    02EC   829          PUSHR   #^M<R5,R6,R8>              ; Save registers
     50   01   D0    02F0   830          MOVL    #1,R0                      ; "Success" flag to NET$DELIVER_CI
        00E6   30    02F3   831          BSBW    NET$DELIVER_CI             ; Build NCB, satisfy DECLSERV request
     0160 8F   BA    02F6   832          POPR    #^M<R5,R6,R8>              ; Restore registers
          DB   11    02FA   833          BRB     10$                        ; Keep looking
               05    02FC   834 30$:     RSB                                ; Done
```

```
                        02FD   836 .SBTTL  NET$STARTUP_OBJ     - Startup privileged process
                        02FD   837 .SBTTL  NET$STARTUP_OBJ_NAM - Startup process by name
                        02FD   838 ;+
                        02FD   839 ;
                        02FD   840 ; Startup a privileged object process if it is not already running.  This is
                        02FD   841 ; used to create EVL for event logging and NML for down-line loading or
                        02FD   842 ; up-line dumping.
                        02FD   843 ;
                        02FD   844 ; Inputs:
                        02FD   845 ;
                        02FD   846 ;       R8    = Object number to start (If NET$STARTUP_OBJ)
                        02FD   847 ;       R7/R8 = Object name to start (If NET$STARTUP_OBJ_NAM)
                        02FD   848 ;
                        02FD   849 ;       R2,R3 = Descriptor of string to be passed as SYS$NET to process
                        02FD   850 ;       R4,R5 = Descriptor of string to be used as process name
                        02FD   851 ;               If =0 then use the object's name as the process name
                        02FD   852 ;
                        02FD   853 ; Outputs:
                        02FD   854 ;
                        02FD   855 ;       R1   PID if process has been created
                        02FD   856 ;       R0   Status
                        02FD   857 ;-
                        02FD   858         .ENABL  LSB
                        02FD   859
                        02FD   860 NET$STARTUP_OBJ_NAM::
     0F80 8F    BB      02FD   861         PUSHR   #^M<R7,R8,R9,R10,R11>   ; Save registers
5B   0000'CF    D0      0301   862         MOVL    NET$GL_CNR_OBI,R11      ; Point to OBI database
        5A      D4      0306   863         CLRL    R10                     ; and start at beginning of list
                        0308   864         $SEARCH eql,obi,l,nam           ; Search for specified object
        18      11      0315   865         BRB     1$                      ; Join common code
                        0317   866
                        0317   867 NET$STARTUP_OBJ::                       ; Startup privileged process
     0F80 8F    BB      0317   868         PUSHR   #^M<R7,R8,R9,R10,R11>   ; Save registers
5B   0000'CF    D0      031B   869         MOVL    NET$GL_CNR_OBI,R11      ; Point to OBI database
        5A      D4      0320   870         CLRL    R10                     ; and start at beginning of list
                        0322   871         $SEARCH eql,obi,l,num           ; Search for specified object
        51      D4      032F   872 1$:      CLRL    R1                      ; Clear PID
0024'CF 52      7D      0331   873         MOVQ    R2,NET_Q_NCB            ; Store descriptor of SYS$NET string
     1C 50      E9      0336   874         BLBC    R0,2$                   ; Skip if not defined as object
                        0339   875         ;
                        0339   876         ;  If object has already declared itself, then it is running
                        0339   877         ;
                        0339   878         $GETFLD obi,l,ucb               ; If UCB NE 0, it has declared itself
     0E 50      E8      0344   879         BLBS    R0,2$                   ; If declared, then its already running
                        0347   880         ;
                        0347   881         ;  If not, get the access control string and process name
                        0347   882         ;
                        0347   883         $GETFLD obi,s,sfi               ; Get the process file name
     03 50      E8      0352   884         BLBS    R0,5$                   ; Skip if specified
      007F      31      0355   885 2$:      BRW     80$                     ; Return with status in R0
0034'CF 57      7D      0358   886 5$:      MOVQ    R7,NET_Q_TSK            ; Save the descriptor
002C'CF 54      7D      035D   887         MOVQ    R4,NET_Q_PRC            ; Setup process name
        10      12      0362   888         BNEQ    10$                     ; If NEQ then name is non-null
                        0364   889         $GETFLD obi,s,nam               ; Else get object name
002C'CF 57      7D      036F   890         MOVQ    R7,NET_Q_PRC            ; Use as process name
003C'CF 05      D0      0374   891 10$:     MOVL    #DET_C_ACC,NET_Q_ACC    ; Setup descriptor of access control
0040'CF 0044'CF 9E      0379   892         MOVAB   DET_AB_ACC,NET_Q_ACC+4  ; data used for create detached,
```

```
                     0380    893                                        ; privileged processes.
                     0380    894                ;
                     0380    895                ;  Start the process with privileges
                     0380    896                ;
                     0380    897                $CREPRC_S       -                   ; create a process
                     0380    898                    INPUT  = NET_Q_TSK,-            ; Network .COM filename
                     0380    899                    OUTPUT = NET_Q_ACC,-            ; Network access control string
                     0380    900                    ERROR  = NET_Q_NCB,-            ; SYS$NET logical name string
                     0380    901                    PRCNAM = NET_Q_PRC,-            ; Process name
                     0380    902                    IMAGE  = NET_Q_IMAGE,-          ; Image (LOGINOUT) to run first
                     0380    903                    PIDADR = NET_L_PID,-            ; Place to store process id
                     0380    904                    MBXUNT = MBX_UNIT,-             ; MBX for termination
                     0380    905                    BASPRI = G^SYS$GB_DEFPRI,-      ; Priority
                     0380    906                    UIC    = #<^O1@16+^O4>,-        ; UIC is [1,4]
                     0380    907                    STSFLG = <#STS_M_NETLOG!-       ; Network login parameters (IN,OUT,ERR)
                     0380    908                              STS_M_NOAUTH!-        ; Use caller's privs/quotas/etc.
                     0380    909                              STS_M_NOACNT>         ; Do not add any accounting records
          18 50      E9  03BC    910                BLBC    R0,80$                  ; If LBC then failed
    50    0004'CF    D0  03BF    911                MOVL    NET_L_PID,R0            ; Get the EPID returned by CREPRC
     00000000'GF     16  03C4    912                JSB     G^EXE$EPID_TO_IPID      ; Convert to internal PID format
0004'CF    50        D0  03CA    913                MOVL    R0,NET_L_PID            ; Use internal format of PID
51    0004'CF        D0  03CF    914                MOVL    NET_L_PID,R1            ; Return the PID to caller
      50    00'      D0  03D4    915                MOVL    S^#SS$_NORMAL,R0        ; Success
      0F80 8F        BA  03D7    916 80$:           POPR    #^M<R7,R8,R9,R10,R11>   ; Restore registers
                     05  03DB    917                RSB
                         03DC    918
                         03DC    919                .DSABL  LSB
```

```
                        03DC    921 .SBTTL  NET$DELIVER_CI  - Process and Deliver Inbound Connect
                        03DC    922 ;++
                        03DC    923 ;
                        03DC    924 ;  A non-zero destination object number indicates that NETACP must fetch the
                        03DC    925 ;  name of the .COM file from the OBJ block - using 'SYS$SYSROOT:[SYSEXE]' as
                        03DC    926 ;  the default directory.  A zero destination object number indicates that the
                        03DC    927 ;  .COM file name is the same as the destination taskname - the default login
                        03DC    928 ;  directory account is assumed to contain the taskname.COM.
                        03DC    929 ;
                        03DC    930 ;       \update this to include tasks with a file i.d.\;!
                        03DC    931 ;
                        03DC    932 ;  If the incoming USER,PSW,ACCT strings are all null, then the default
                        03DC    933 ;  inbound access control for the specified object (or task) are used (these
                        03DC    934 ;  strings may also be null).  This allows a DECnet-VAX node to serve as a
                        03DC    935 ;  convenient host particularly for RSX-11S.
                        03DC    936 ;
                        03DC    937 ;  This routines determines whether the connect is to be handed to a task
                        03DC    938 ;  which has declared a name or an object type.
                        03DC    939 ;
                        03DC    940 ;
                        03DC    941 ;  INPUTS:       R11     LLI CNR address (if low bit set in R0)
                        03DC    942 ;                R10     LLI CNF address (if low bit set in R0)
                        03DC    943 ;                R6      XWB address
                        03DC    944 ;                R0      Low bit set    =>  deliver connect notification
                        03DC    945 ;                        Low bit clear  =>  tell NETDRIVER that resource error
                        03DC    946 ;                                           occurred
                        03DC    947 ;
                        03DC    948 ;  OUTPUTS:      R11,R10,R6 are preserved.
                        03DC    949 ;
                        03DC    950 ;                All other registers are clobbered.
                        03DC    951 ;
                        03DC    952 ;
                        03DC    953 ;  SIDE EFFECTS: Process created if needed, image started
                        03DC    954 ;
                        03DC    955 ;--
                        03DC    956
                        03DC    957 ;
                        03DC    958 ;  Define scratch storage
                        03DC    959 ;
                        03DC    960
                0000000C 03DC   961 ACC =  12                               ; Composite access strings
                000000C8 03DC   962 PRC = 200                               ; Process name
                0000012C 03DC   963 TSK = 300                               ; Image to run
                000003E8 03DC   964 CONN_SPACE = 1000                       ; Size of scratch storage
                         03DC   965
                         03DC   966 NET$DELIVER_CI:
          0018'CF    D4  03DC   967         CLRL    PTR_NCB_BUF            ; No NCB buffer yet
          001C'CF    D4  03E0   968         CLRL    PTR_CON_BUF            ; No scratch buffer yet
                         03E4   969         ;
                         03E4   970         ;   Initialize parameters for call to NETDRIVER
                         03E4   971         ;
  000C'CF    3E A6    3C  03E4   972         MOVZWL  XWB$W_LOCLNK(R6),NET_L_LNK      ; Setup logical link address
     0000'CF    01    9A  03EA   973         MOVZBL  #NETUPD$_ABORT,  NET_L_FCT      ; Assume process couldn't start
0014'CF    0000'CF    D0  03EF   974         MOVL    NET$GL_NET_UCB,  NET_L_UCB      ; Default is our UCB
     0020'CF    5A    D0  03F6   975         MOVL    R10,             NET_A_LLI      ; Save LLI pointer
          0004'CF    D4  03FB   976         CLRL                     NET_L_PID      ; No PID yet
          0010'CF    D4  03FF   977         CLRL                     NET_A_NCB      ; No NCB yet
```

```
                                0403   978          ;
                                0403   979          ;
                                0403   980          ;
                     57 50   E9 0403   981          BLBC    R0,3$                   ; If LBC, resource error encountered
                                0406   982                                          ; by caller
                51   03E8 8F   3C 0406   983          MOVZWL  #CONN_SPACE,R1          ; Set size of scratch buffer
                       FBF2'   30 040B   984          BSBW    NET$ALLOCATE            ; Allocate a scratch buffer
                     4C 50   E9 040E   985          BLBC    R0,3$                   ; Br if allocation failure, notify
                                0411   986                                          ;  driver
                001C'CF   5C   D0 0411   987          MOVL    R2,PTR_CON_BUF          ; Save address for deallocation
                                0416   988          ;
                                0416   989          ;    Initialize descriptors and data for process creation
                                0416   990          ;
                  53   0C A2   9E 0416   991          MOVAB   ACC(R2),R3              ; Get ACC address
                0040'CF   53   D0 041A   992          MOVL    R3,NET_Q_ACC+4          ; Store it
                003C'CF   53   CE 041F   993          MNEGL   R3,NET_Q_ACC            ; Bias ACC size
                53   00C8 C2   9E 0424   994          MOVAB   PRC(R2),R3              ; Get PRC address
                0030'CF   53   D0 0429   995          MOVL    R3,NET_Q_PRC+4          ; Store it
                002C'CF   53   CE 042E   996          MNEGL   R3,NET_Q_PRC            ; Bias PRC size
                53   012C C2   9E 0433   997          MOVAB   TSK(R2),R3              ; Get TSK address
                0038'CF   53   D0 0438   998          MOVL    R3,NET_Q_TSK+4          ; Store it
                0034'CF   53   CE 043D   999          MNEGL   R3,NET_Q_TSK            ; Bias TSK size
                                0442  1000          ;
                                0442  1001          ;    Set default values
                                0442  1002          ;
                  50   0000'CF   D0 0442  1003          MOVL    NET$GL_PTR_VCB,R0               ; Point to RCB
              0049'CF   67 A0   90 0447  1004          MOVB    RCB$B_ECL_DPX(R0),OBI_B_PRX     ; Set default OBI proxy access
                 004A'CF   03   90 044D  1005          MOVB    #NMA$C_ACES_BOTH, INT_B_PRX     ; Set default internal proxy
                                0452  1006                                                  ; access state
                                0452  1007          ;
                                0452  1008          ;    Allocate scratch buffer from nonpaged pool for NCB
                                0452  1009          ;
           51   007B 8F   3C 0452  1010          MOVZWL  #NET$C_MAX_NCB+13,R1            ; Length of buffer for an NCB
                   FBA6'   30 0457  1011          BSBW    NET$ALONPAGED                   ; Allocate the buffer
                    17 50   E8 045A  1012          BLBS    R0,5$                           ; If LBS then block allocated
                                045D  1013          ;
                                045D  1014          ;   Tell NETDRIVER about resource error
                                045D  1015          ;
           0008'CF   01   9A 045D  1016 3$:       MOVZBL  #NET$C_DR_RSU,NET_L_REASON      ; Reason is "resource error"
           50   0000'CF   D0 0462  1017          MOVL    NET$GL_PTR_VCB,R0               ; Get RCB pointer
                                0467  1018          BUMP    W^RCB$W_CNT_XRE(R0)             ; Account for resource error
                        34   11 0472  1019          BRB     10$                             ; Continue
                                0474  1020          ;
                                0474  1021          ;  Build the NCB and locate the process to accept it
                                0474  1022          ;
           0018'CF   52   D0 0474  1023 5$:       MOVL    R2,PTR_NCB_BUF                  ; Save for deallocation
             53   0D A2   9E 0479  1024          MOVAB   13(R2),R3                       ; Get address of string, leave
                                047D  1025                                                  ; room for count and buf header
           0028'CF   53   D0 047D  1026          MOVL    R3,NET_Q_NCB+4                  ; Store it
           0024'CF   53   CE 0482  1027          MNEGL   R3,NET_Q_NCB                    ; Bias NCB size
                      0032   30 0487  1028          BSBW    BUILD_NCB                       ; Build the NCB
                     1B 50   E9 048A  1029          BLBC    R0,10$                          ; If LBC then error
0000006E 8F   0024'CF   D1 048D  1030          CMPL    NET_Q_NCB,#NET$C_MAX_NCB        ; Make sure we didn't write
                                0496  1031                                                  ; past end of buffer
                                0496  1032          ASSUME  NET$C_MAX_NCB  LE  255          ; Must allow counted string fmt
           50   0028'CF   D0 0496  1033          MOVL    NET_Q_NCB+4,R0                  ; Get ptr to NCB
           70   0024'CF   90 049B  1034          MOVB    NET_Q_NCB,-(R0)                 ; Enter count field and
```

```
                0010'CF    50    DO   04A0  1035          MOVL    RO,NET_A_NCB                    ; save its address in case NCB
                                      04A5  1036                                                ; is to be passed to NETDRIVER
                                      04A5  1037                                                ; for a declared name
                     00B2  30    04A5  1038          BSBW    GET_PROC                        ; Find/create process to
                                      04A8  1039                                                ; receive the connect
                     0440  30    04A8  1040 10$:     BSBW    TELL_DRV                        ; Tell driver about connect
                50   0018'CF    DO   04AB  1041          MOVL    PTR_NCB_BUF,RO                  ; Address of buffer
                     FB4D'  30    04B0  1042          BSBW    NET$DEALLOCATE                  ; Deallocate the buffer
                50   001C'CF    DO   04B3  1043          MOVL    PTR_CON_BUF,RO                  ; Address of scratch buffer
                     FB45'  30    04B8  1044          BSBW    NET$DEALLOCATE                  ; Deallocate scratch storage
                            05    04BB  1045          RSB                                     ; Done
                                      04BC  1046
```

```
                        04BC  1048 .SBTTL  BUILD_NCB       - Build NCB for incoming connect
                        04BC  1049 ;+
                        04BC  1050 ;
                        04BC  1051 ; This routine builds the NCB string for the connect, to be later
                        04BC  1052 ; given to the destination process (in any number of different ways).
                        04BC  1053 ;
                        04BC  1054 ; Inputs:
                        04BC  1055 ;
                        04BC  1056 ;       R6 = XWB address
                        04BC  1057 ;       NET_Q_NCB = Descriptor of scratch space for NCB
                        04BC  1058 ;
                        04BC  1059 ; Outputs:
                        04BC  1060 ;
                        04BC  1061 ;       R0 = status code
                        04BC  1062 ;       NET_Q_NCB = Descriptor of resultant NCB
                        04BC  1063 ;-
                        04BC  1064 BUILD_NCB:                                   ; Build the NCB
                        04BC  1065         ;
                        04BC  1066         ;    Enter 'nodename::'
                        04BC  1067         ;
    53   0028'CF    D0  04BC  1068         MOVL    NET_Q_NCB+4,R3              ; Get output buffer pointer
    5B   0000'CF    D0  04C1  1069         MOVL    NET$GL_CNR_NDI,R11          ; Get root for search
           5A       D4  04C6  1070         CLRL    R10                         ; Indicate no NDI yet
      58    3A A6    3C  04C8  1071         MOVZWL  XWB$W_REMNOD(R6),R8         ; Get remote node address
                        04CC  1072         $SEARCH eql,ndi,l,tad               ; Find NDI with matching address
           18 50    E9  04D9  1073         BLBC    R0,10$                      ; If LBC none, use node address
                        04DC  1074         $GETFLD ndi,s,nna                   ; Get the node name
           0A 50    E9  04E7  1075         BLBC    R0,10$                      ; Invalid if LBC
              57    95  04EA  1076         TSTB    R7                          ; Is name null ?
              06    13  04EC  1077         BEQL    10$                         ; If EQL use node address
                        04EE  1078         ;
                        04EE  1079         ;    Enter ASCII nodename
                        04EE  1080         ;
    63   68   57    28  04EE  1081         MOVC3   R7,(R8),(R3)                ; Move node name
              07    11  04F2  1082         BRB     20$
                        04F4  1083         ;
                        04F4  1084         ;    Enter node address converted to ASCII
                        04F4  1085         ;
      50    3A A6    3C  04F4  1086 10$:    MOVZWL  XWB$W_REMNOD(R6),R0         ; Get node address
              FB05'    30  04F8  1087         BSBW    NET$BIN2ASC                 ; Move after conversion to ASCII
    83   3A3A 8F    B0  04FB  1088 20$:    MOVW    #^A'::',(R3)+               ; Move delimiter
                        0500  1089         ;
                        0500  1090         ;    Enter taskname
                        0500  1091         ;
           83   22    90  0500  1092         MOVB    #^A'"',(R3)+                ; Enter delimiter
    50   00BA C6    9A  0503  1093         MOVZBL  XWB$T_RPRNAM+1(R6),R0       ; Get object number
              08    13  0508  1094         BEQL    30$                         ; If EQL then use taskname
              FAF3'    30  050A  1095         BSBW    NET$BIN2ASC                 ; Else convert to ASCII and move
           83   3D    90  050D  1096         MOVB    #^A'=',(R3)+                ; Enter delimiter
              19    11  0510  1097         BRB     50$                         ; Continue
                        0512  1098         ;
                        0512  1099         ;    Enter 0=taskname
                        0512  1100         ;
    83   3D30 8F    B0  0512  1101 30$:    MOVW    #^A'0=',(R3)+               ; Enter 0=
    51   00B9 C6    9E  0517  1102         MOVAB   XWB$T_RPRNAM(R6),R1         ; Point to process name field
              53    DD  051C  1103         PUSHL   R3                          ; Save pointer
             0378    30  051E  1104         BSBW    GET_PR_NAM                  ; Move the name text
```

```
                          53 8ED0  0521  1105          POPL    R3                      ; Recover pointer
                       32 50   E9  0524  1106          BLBC    R0,60$                  ; If LBC then illegal name format
                 63    68   57   28  0527  1107          MOVC3   R7,(R8),(R3)            ; Enter taskname
                                    052B  1108          ;
                                    052B  1109          ;    Enter remainder of NCB
                                    052B  1110          ;
                          83   2F   90  052B  1111 50$:    MOVB    #^A'/',(R3)+            ; Enter delimiter
                       83 000C'CF   B0  052E  1112          MOVW    NET_L_LNK,(R3)+         ; Enter local link number
                         51   5B A6   9E  0533  1113          MOVAB   XWB$B_DATA(R6),R1       ; Get address of counted data
                          50   61   9A  0537  1114          MOVZBL  (R1),R0                 ; Get its length
                               50   B6  053A  1115          INCW    R0                      ; Include its count field
                63    11   00   61   50  2C  053C  1116          MOVC5   R0,(R1),#0,#17,(R3)     ; Enter into fixed size field
                          51 00A4   C6  9E  0542  1117          MOVAB   XWB$B_LPRNAM(R6),R1     ; Address local task specifier
                          50   81   9A  0547  1118          MOVZBL  (R1)+,R0                ; Get its length
                          63   61   50  28  054A  1119          MOVC3   R0,(R1),(R3)            ; Move it
                          83   22   90  054E  1120          MOVB    #^A'"',(R3)+            ; Enter terminator
                    0024'CF    53   C0  0551  1121          ADDL    R3,NET_Q_NCB            ; Update size in descriptor
                          50   01   90  0556  1122          MOVB    #1,R0                   ; Indicate success
                                    05  0559  1123 60$:    RSB
```

```
                     055A  1125 .SBTTL  GET_PROC        - Locate process to accept connect
                     055A  1126 ;+
                     055A  1127 ;
                     055A  1128 ; Find the OBI block associated with the local object.  If the OBI is
                     055A  1129 ; for a declared name or object then pass the NCB to the declaring
                     055A  1130 ; process's mailbox, otherwise create a process to receive the connect.
                     055A  1131 ; If there is a server process waiting for more work, then tell the
                     055A  1132 ; server process that it can have the connect request.
                     055A  1133 ;
                     055A  1134 ; Inputs:
                     055A  1135 ;
                     055A  1136 ;       R6 = XWB address
                     055A  1137 ;
                     055A  1138 ;       Own storage
                     055A  1139 ;
                     055A  1140 ; Outputs:
                     055A  1141 ;
                     055A  1142 ;       None
                     055A  1143 ;-
                     055A  1144 GET_PROC:                                    ; Get process to accept the connect
5B   0000'CF   D0    055A  1145         MOVL    NET$GL_CNR_OBI,R11          ; Set up OBI CNR
51   00A5 C6   9E    055F  1146         MOVAB   XWB$T_LPRNAM(R6),R1         ; Address local task specifier
        033C   30    0564  1147         BSBW    GET_PR_ZNA                  ; Get its ZNA field
          2B 50    E9 0567  1148         BLBC    R0,TO$                      ; If LBC then format error
                     056A  1149         ;
                     056A  1150         ;
                     056A  1151         ;   Find the OBI CNF
                     056A  1152         ;
                     056A  1153         ;
          04   3C    056A  1154         MOVZWL  #NET$C_DR_NOBJ,-            ; Assume failure due to unknown object
     0008'CF         056C  1155                 NET_L_REASON                ;
          5A   D4    056F  1156         CLRL    R10                         ; Indicate no current CNF
                     0571  1157         $SEARCH eql,obi,s,zna               ; Find OBI block with this CNF
        17 50    E8 057E  1158         BLBS    R0,20$                      ; If LBS then CNF was found
          68   95    0581  1159         TSTB    (R8)                        ; Is this a numbered object connect ?
          10   12    0583  1160         BNEQ    10$                         ; If NEQ then no such object
57   000B'CF   7D    0585  1161         MOVQ    NET_Q_TASKZNA,R7            ; Else use default TASK  ZNA descriptor
     51   00   9A    058A  1162         MOVZBL  S^#NFB$C_OP_EQL,R1          ; Specify match operator
          5A   D4    058D  1163         CLRL    R10                         ; Start from head of list
        FA6E'  30    058F  1164         BSBW    CNF$KEY_SEARCH              ; Look for the CNF
        3A 50  E8    0592  1165         BLBS    R0,25$                      ; If LBS then found, br to continue
         021B  31    0595  1166 10$:     BRW     100$                        ; Complete with error
                     0598  1167         ;
                     0598  1168         ;
                     0598  1169         ;   The OBI CNF has been found.  See if the object has been "declared"
                     0598  1170         ;   If not, build the .COM file file i.d. and setup its descriptor.
                     0598  1171         ;
                     0598  1172         ;
                     0598  1173 20$:     $GETFLD obi,l,ucb                   ; Get the associated UCB
        52 50    E9 05A3  1174         BLBC    R0,30$                      ; If LBC then not declared name
0014'CF    58    D0 05A6  1175         MOVL    R8,NET_L_UCB                ; Save the UCB pointer
                     05AB  1176         $GETFLD obi,l,pid                   ; Get the declarer's EPID
        3F 50    E9 05B6  1177         BLBC    R0,30$                      ; If LBC then treat as undeclared
     50    58    D0 05B9  1178         MOVL    R8,R0                       ; Conver. from EPID to IPID
 00000000'GF   16    05BC  1179         JSB     G^EXE$EPID_TO_IPID          ; ...
0004'CF    50    D0 05C2  1180         MOVL    R0,NET_L_PID                ; Save the PID
          02   9A    05C7  1181         MOVZBL  #NETUPD$_CONNECT,-          ; Setup the function code
```

```
                0000'CF        05C9  1182                     NET_L_FCT               ;
                   01E4   31   05CC  1183          BRW      100$                    ; Return to pass NCB to mailbox
                               05CF  1184          ;
                               05CF  1185          ;   The object is a named object which could not be found in the
                               05CF  1186          ;   object database.  Use the requested object name to construct
                               05CF  1187          ;   the name of the command procedure, rather than consulting the
                               05CF  1188          ;   OBI entry (we are currently set to the "TASK" OBI).  If the
                               05CF  1189          ;   object name starts with a "$", then the object is "reserved
                               05CF  1190          ;   to DEC", and we get the command procedure from SYS$SYSTEM.
                               05CF  1191          ;
          51    00A5 C6    9E  05CF  1192 25$:     MOVAB    XWB$T_LPRNAM(R6),R1    ; Address local task specifier
                   02C2   30   05D4  1193          BSBW     GET_PR_NAM             ; Get its name
          53    0038'CF    D0  05D7  1194          MOVL     NET_Q_TSK+4,R3         ; Get address of output buffer
                24   68   91  05DC  1195          CMPB     (R8),#^A"$"            ; Does the name start with "$"?
                     0C   12  05DF  1196          BNEQ     28$                    ; If so,
                     58   D6  05E1  1197          INCL     R8                     ; Strip "$" off front of name
                     57   D7  05E3  1198          DECL     R7
                0032'CF   28  05E5  1199          MOVC     NET_Q_SYSTEM,-         ; Prefix name with "SYS$SYSTEM:"
          63    0036'DF        05E9  1200                   @NET_Q_SYSTEM+4,(R3)
          63    68   57   28  05ED  1201 28$:     MOVC3    R7,(R8),(R3)           ; Move the name
          0034'CF    53   C0  05F1  1202          ADDL     R3,NET_Q_TSK           ; Update filename size
                     26   11  05F6  1203          BRB      40$                    ; Continue
                               05F8  1204          ;
                               05F8  1205          ;   Build filespec of object command procedure
                               05F8  1206          ;
                     04   3C  05F8  1207 30$:     MOVZWL   #NET$C_DR_NOBJ,-       ; Assume error
                0008'CF        05FA  1208                   NET_L_REASON           ;
                               05FD  1209          $GETFLD obi,s,sfi               ; Get parsed file id
                   6B 50   E9  0608  1210          BLBC     R0,55$                 ; If LBC then file id is invalid
          0034'CF    57   7D  060B  1211          MOVQ     R7,NET_Q_TSK           ; Update filename descriptor
                               0610  1212          ;
                               0610  1213          ;
                               0610  1214          ;   Create a process name.
                               0610  1215          ;
                               0610  1216          ;
                               0610  1217          $GETFLD obi,s,nam               ; Get object name for prefix
                      05 50    E8  061B  1218          BLBS     R0,50$                 ; If LBS then name was found
               57     0000'CF  7D  061E  1219 40$:     MOVQ     NET_Q_NETPREFIX,R7     ; Setup standard prefix descriptor
          0030'DF     68   57  28  0623  1220 50$:     MOVC3    R7,(R8),@NET_Q_PRC+4   ; Move the prefix
                83    5F   8F  90  0629  1221          MOVB     #^A' ',(R3)+           ; Move the delimiter
               50     000C'CF  D0  062D  1222          MOVL     NET_L_LNK,R0           ; Get the local link number
                         F9CB' 30  0632  1223          BSBW     NET$BIN2ASC            ; Convert to ascii and append as
                               0635  1224                                           ; the suffix
               002C'CF    53   C0  0635  1225          ADDL     R3,NET_Q_PRC           ; Done with process name
                               063A  1226          ;
                               063A  1227          ;   If the connect did not use format type 2, then don't attempt
                               063A  1228          ;   a proxy login.
                               063A  1229          ;
               02    00B9 C6   91  063A  1230          CMPB     XWB$T_RPRNAM(R6),#2    ; Format type 2?
                          05   13  063F  1231          BEQL     51$                    ; Branch if so
               004A'CF    00   90  0641  1232          MOVB     #NMA$C_ACES_NONE,INT_B_PRX ; Disallow proxy access
                               0646  1233 51$:     ;
                               0646  1234          ;
                               0646  1235          ;   If no access control was specified, use default from OBI block
                               0646  1236          ;
                               0646  1237          ;
                               0646  1238          $GETFLD obi,l,prx               ; Get proxy login state
```

```
                05 50    E9  0651  1239          BLBC    R0,52$                  ; If LBC then none specified
          0049'CF  58    90  0654  1240          MOVB    R8,OBI_B_PRX            ; Store it
58        00C6  6     9E  0659  1241 52$:        MOVAB   XWB$B_LOGIN(R6),R8      ; Get address of access info
          57  58    9A  065E  1242          MOVZBL  (R8)+,R7                ; Get total size
          03  57    91  0661  1243          CMPB    R7,#3                   ; Is it 3 null (counted) strings
                13  13  0664  1244          BEQL    60$                     ; If so use access info in OBI
                00  90  0666  1245          MOVB    #NMA$C_ACES_NONE,-      ; Disallow proxy access
            004A'CF     0668  1246                      INT_B_PRX           ; Store it
             75 8F  57  91  066B  1247          CMPB    R7,#NET$C_MAXACCFLD*3   ; Too long ?
                13  1B  066F  1248          BLEQU   70$                     ; If LEQU then move the strings
                2B  3C  0671  1249          MOVZWL  #NET$C_DR_IMLONG,-      ; Indicate network failure type
            0008'CF     0673  1250                      NET_L_REASON        ;
               013A  31  0676  1251 55$:        BRW     100$                    ; Continue
                         0679  1252                      ;
                         0679  1253 60$:        $GETFLD obi,s,iac               ; Get inbound access control
                         0684  1254 70$:        ;
                         0684  1255             ;
                         0684  1256             ;  Enter the flags word followed by the access control strings
                         0684  1257             ;
                         0684  1258             ;
53        0040'CF   D0  0684  1259          MOVL    NET_Q_ACC+4,R3          ; Get pointer to access control buffer
                83  B4  0689  1260          CLRW    (R3)+                   ; Clear the flags word
                         068B  1261          $DISPATCH  TYPE=B,INT_B_PRX -   ; Don't set flag if proxy disallowed
                         068B  1262          <-
                         068B  1263            <NMA$C_ACES_OUTG, 80$>-
                         068B  1264            <NMA$C_ACES_NONE, 80$>-
                         068B  1265          >
                         0697  1266          $DISPATCH  TYPE=B,OBI_B_PRX -   ; Don't set flag if proxy disallowed
                         0697  1267          <-
                         0697  1268            <NMA$C_ACES_OUTG, 80$>-
                         0697  1269            <NMA$C_ACES_NONE, 80$>-
                         0697  1270          >
             FE A3  01  A8  06A3  1271          BISW    #1,-2(R3)               ; Say "proxy login allowed"
63           68  57  28  06A7  1272 80$:        MOVC3   R7,(R8),(R3)            ; Move access control strings,
                         06AB  1273                                          ; even if it's null
003C'CF          53  C0  06AB  1274          ADDL    R3,NET_Q_ACC            ; Complete string size calc.
                024B  30  06B0  1275          BSBW    UP_CASE                 ; Up-case all pertinent strings
                         06B3  1276          ;
                         06B3  1277          ;  Attempt to find an available server process which is waiting
                         06B3  1278          ;  for a connect which matches it's context.
                         06B3  1279          ;
5B        0000'CF   D0  06B3  1280          MOVL    NET$GL_CNR_SPI,R11      ; Get root of SPI database
                5A  D4  06B8  1281          CLRL    R10                     ; Start at beginning of list
                58  D4  06BA  1282 81$:        CLRL    R8                      ; Search key is zero
                         06BC  1283          $SEARCH neq,spi,l,irp           ; Find an SPI with an IRP NE 0
                03 50   E8  06CA  1284          BLBS    R0,82$                  ; Br if found, check process
                0082  31  06CD  1285          BRW     89$                     ; Else, create process
                34 A6   D5  06D0  1286 82$:        TSTL    XWB$L_PID(R6)           ; Is this connect "tagged" for a
                   14   13  06D3  1287          BEQL    83$                     ; specific process?
                         06D5  1288          $GETFLD spi,l,pid               ; If so, get PID of this server
                D7 50   E9  06E0  1289          BLBC    R0,81$                  ; (if not present, error, skip entry)
             34 A6  58  D1  06E3  1290          CMPL    R8,XWB$L_PID(R6)        ; Is this server the intended process?
                D1  12  06E7  1291          BNEQ    81$                     ; If not, then continue searching
                         06E9  1292 83$:        ;
                         06E9  1293             ;  Always check the access control, even for processes started
                         06E9  1294             ;  with proxy requested.  This way, if different default access
                         06E9  1295             ;  control is used (each object can specify a unique account,
```

```
                                06E9  1296          ;   including NONE), the wrong process isn't matched.
                                06E9  1297          ;
                                06E9  1298          $GETFLD spi,s,acs               ; Get ACS for server process
                             C3 50    E9  06F4  1299          BLBC    R0,81$          ; (if not present, error, skip entry)
                          50 003C'CF  7D  06F7  1300          MOVQ    NET_Q_ACC,R0    ; Get access string for new connect
          61    50    00  68    57    2D  06FC  1301          CMPC5   R7,(R8),#0,R0,(R1) ; Does it match?
                             B6    12  0702  1302          BNEQ    81$             ; If no match, keep searching
                                0704  1303          ;
                                0704  1304          ;   Make sure the process's "proxy request" flag matches.
                                0704  1305          ;
                                0704  1306          $GETFLD spi,v,prl               ; Get proxy login flag
                             A8 50    E9  070F  1307          BLBC    R0,81$          ; (if not present, error, skip entry)
          58    0040'DF    01    00    ED  0712  1308          CMPZV   #0,#1,@NET_Q_ACC+4,R8 ; Does proxy login flag match?
                             9F    12  0719  1309          BNEQ    81$             ; If not, try to find another server
                                071B  1310          ;
                                071B  1311          ;   For logical links which request proxy access, require
                                071B  1312          ;   that the requesting node and username match as well.
                                071B  1313          ;
                             2F 58    E9  071B  1314          BLBC    R8,87$          ; If proxy requested,
                                071E  1315          $GETFLD spi,l,rna               ; Get remote node address for server
                             8E 50    E9  0729  1316          BLBC    R0,81$          ; (if not present, error, skip entry)
                       58    3A A6    B1  072C  1317          CMPW    XWB$W_REMNOD(R6),R8 ; Is it the same node as the connect?
                             88    12  0730  1318          BNEQ    81$             ; If not, try to find another server
                                0732  1319          $GETFLD spi,s,rid               ; Get remote user ID for server
                             0F 50    E9  073D  1320          BLBC    R0,88$          ; (if not present, error, skip entry)
                          50    6F A6    9A  0740  1321          MOVZBL  XWB$B_RID(R6),R0 ; Get length of RID for new connect
      70 A6    50    00    68    57    2D  0744  1322          CMPC5   R7,(R8),#0,R0,XWB$T_RID(R6) ; Does it match?
                             02    12  074B  1323          BNEQ    88$             ; If no match, then skip it
                             65    11  074D  1324  87$:     BRB     SEND_TO_SERVER  ; Server ok, send it the connect
                                074F  1325
                          FF68    31  074F  1326  88$:     BRW     81$             ; (Branch helper to top of loop)
                                0752  1327  89$:     ;
                                0752  1328          ;
                                0752  1329          ;   Create the user process
                                0752  1330          ;
                                0752  1331          $CREPRC_S       -               ; create a process
                                0752  1332              INPUT=  NET_Q_PROC,-        ; Network NETSERVER.COM filename
                                0752  1333              OUTPUT= NET_Q_ACC,-         ; Access control strings
                                0752  1334              ERROR=  NET_Q_NCB,-         ; 1st NCB (solely for LOGIN proxy use)
                                0752  1335              PRCNAM= NET_Q_PRC,-         ; Process name
                                0752  1336              IMAGE=  NET_Q_IMAGE,-       ; Image (LOGINOUT) to run first
                                0752  1337              PIDADR= NET_L_PID,-         ; Place to store process id
                                0752  1338              BASPRI= G^SYS$GB_DEFPRI,-   ; Priority
                                0752  1339              UIC=    #<^O10@16+^O40>,-   ; UIC is [10,40]
                                0752  1340              STSFLG= #<STS_M_NETLOG>,-   ; This is a network process
                                0752  1341              MBXUNT= MBX_UNIT            ; MBX for termination
                                078E  1342                                          ; notification
                          07 50    E8  078E  1343          BLBS    R0,90$          ; If LBS process was created
                             01    3C  0791  1344          MOVZWL  #NET$C_DR_RSU,-     ; Assume because couldn't get
                       0008'CF          0793  1345                  NET_L_REASON    ; the resources
                             1B    11  0796  1346          BRB     100$            ; Take common exit
          50    0004'CF    D0  0798  1347  90$:     MOVL    NET_L_PID,R0    ; Get the EPID returned by CREPRC
                58    50    D0  079D  1348          MOVL    R0,R8           ; Save EPID
           00000000'GF    16  07A0  1349          JSB     G^EXE$EPID_TO_IPID      ; Convert to internal PID format
      0004'CF    50    D0  07A6  1350          MOVL    R0,NET_L_PID    ; Use internal format of PID
                   04    3C  07AB  1351          MOVZWL  #NETUPD$_PROCRE,-       ; Say "process created"
                0000'CF          07AD  1352                  NET_C_FCT       ;
```

```
                     07B0  1353          ;
                     07B0  1354          ;    The network process is created.  Now create an SPI database entry
                     07B0  1355          ;    so we can keep track of it.
                     07B0  1356          ;
0081   30            07B0  1357          BSBW    CREATE_SPI                  ; Create SPI database entry
                     07B3  1358                                              ; Ignore errors if can't be inserted
       05            07B3  1359 100$:    RSB                                 ; Common exit
```

```
                        07B4  1361 .SBTTL  SEND_TO_SERVER  - Send connect to waiting server
                        07B4  1362 ;+
                        07B4  1363 ;
                        07B4  1364 ; There is a waiting server which can handle the incoming connect.  Set
                        07B4  1365 ; it up so that the server can accept the logical link.
                        07B4  1366 ;
                        07B4  1367 ; Inputs:
                        07B4  1368 ;
                        07B4  1369 ;       R11 = SPI CNR address
                        07B4  1370 ;       R10 = CNF for server database entry
                        07B4  1371 ;
                        07B4  1372 ;-
                        07B4  1373 SEND_TO_SERVER:
        56    5A    D0  07B4  1374         MOVL    R10,R6                  ; Save address of old CNF
           F846'    30  07B7  1375         BSBW    NET$GETUTLBUF           ; Get permission tc use utility buffer
           F843'    30  07BA  1376         BSBW    CNF$INIT_UTL            ; Initialize utility buffer
        58    56    D0  07BD  1377         MOVL    R6,R8                   ; Pass address of old CNF
           F83D'    30  07C0  1378         BSBW    CNF$COPY                ; Copy old CNF to new CNF space
57      0024'CF    7D  07C3  1379         MOVQ    NET_Q_NCB,R7            ; Get descriptor of NCB
                        07C8  1380         $PUTFLD spi,s,ncb               ; Store it
57      0034'CF    7D  07D3  1381         MOVQ    NET_Q_TSK,R7            ; Get procedure filespec
                        07D8  1382         $PUTFLD spi,s,sfi               ; Store it
57      002C'CF    7D  07E3  1383         MOVQ    NET_Q_PRC,R7            ; Get process name
                        07E8  1384         $PUTFLD spi,s,pnm               ; Store it
           F80A'    30  07F3  1385         BSBW    CNF$INSERT              ; Insert new CNF (R10 = UTILBUF)
                        07F6  1386                                         ; and delete old CNF (R6)
                        07F6  1387                                         ; returns: R10 = valid CNF
                        07F6  1388         $GETFLD spi,l,pid               ; Get PID of server process
0004'CF    58    D0  0801  1389         MOVL    R8,NET_L_PID            ; Make it seem as if it was just created
                        0806  1390         $GETFLD spi,l,irp               ; Get waiting DECLSERV IRP
           F7EC'    30  0811  1391         BSBW    CNF$CLR_FIELD           ; and clear it from database
        53    58    D0  0814  1392         MOVL    R8,R3                   ; Get IRP address
     38 A3    00'    D0  0817  1393         MOVL    S^#SS$_NORMAL,IRP$L_IOST1(R3) ; Set success into IRP
3C A3    0004'CF    D0  081B  1394         MOVL    NET_L_PID,IRP$L_IOST2(R3) ; Return IPID of SPI process as well
     55    1C A3    D0  0821  1395         MOVL    IRP$L_UCB(R3),R5        ; Get UCB address
  00000000'GF    16  0825  1396         JSB     G^COM$POST              ; and complete the request
           F7D2'    30  082B  1397         BSBW    NET$DEC_TRANS           ; Account for completed transaction
   0000'CF    04    3C  082E  1398         MOVZWL  #NETUPD$_PROCRE,NET_L_FCT ; Tell NETDRIVER that process exists
                    05  0833  1399         RSB
```

```
                                0834  1401 .SBTTL  CREATE_SPI      - Create SPI database entry
                                0834  1402 ;+
                                0834  1403 ;
                                0834  1404 ; Subroutine to create an SPI database entry after having just created
                                0834  1405 ; the network process.
                                0834  1406 ;
                                0834  1407 ; Inputs:
                                0834  1408 ;
                                0834  1409 ;       R6 = XWB address
                                0834  1410 ;       Own storage
                                0834  1411 ;
                                0834  1412 ; Outputs:
                                0834  1413 ;
                                0834  1414 ;       R0 = Status code
                                0834  1415 ;-
                                0834  1416 CREATE_SPI:
                    F7C9'  30   0834  1417         BSBW    NET$GETUTLBUF           ; Get permission to use utility buffer
               5B   0000'CF  D0   0837  1418         MOVL    NET$GL_CNR_SPI,R11      ; Get root of SPI database
                    F7C1'  30   083C  1419         BSBW    CNF$INIT_UTL            ; Init utility buffer as a CNF block
               58   0004'CF  D0   083F  1420         MOVL    NET_L_PID,R8            ; Get PID of server process
                                0844  1421         $PUTFLD spi,l,pid               ; Store parameter into entry
58  0040'DF    01    00      EF   084F  1422         EXTZV   #0,#1,@NET_Q_ACC+4,R8   ; Get proxy flag sent to LOGIN
                                0856  1423         $PUTFLD spi,v,prl               ; Store it
               57   003C'CF  7D   0861  1424         MOVQ    NET_Q_ACC,R7            ; Get access control sent to LOGIN
                                0866  1425         $PUTFLD spi,s,acs               ; Store ACS string sent to LOGIN
                 58   3A A6  3C   0871  1426         MOVZWL  XWB$W_REMNOD(R6),R8     ; Get remote node address
                                0875  1427         $PUTFLD spi,l,rna               ; Store it
                 57   6F A6  9A   0880  1428         MOVZBL  XWB$B_RID(R6),R7        ; Make descriptor of RID
                 58   70 A6  9E   0884  1429         MOVAB   XWB$T_RID(R6),R8
                                0888  1430         $PUTFLD spi,s,rid               ; Store it
                        56    D4   0893  1431         CLRL    R6                      ; No "old" CNF entry
                     F768'  30   0895  1432         BSBW    CNF$INSERT              ; Insert into database
                            05   0898  1433         RSB
```

```
                     0899  1435 .SBTTL  GET_PR_NAM      - Get name of object procedure
                     0899  1436 .SBTTL  GET_PR_ZNA      - Construct ZNA string for an object
                     0899  1437 ;+
                     0899  1438 ;
                     0899  1439 ; Inputs:
                     0899  1440 ;
                     0899  1441 ;       R1 = Address of local task specifier
                     0899  1442 ;
                     0899  1443 ; Outputs:
                     0899  1444 ;
                     0899  1445 ;       R7/R8 = Descriptor of resultant string
                     0899  1446 ;-
                     0899  1447
                     0899  1448                      .ENABL  LSB
                     0899  1449 GET_PR_NAM:
58  011C'CF   9E     0899  1450          MOVAB   ZNABUF,R8                   ; Get procedure name
    53   58   D0     089E  1451          MOVL    R8,R3                       ; Setup buffer pointer
         0C   11     08A1  1452          BRB     5$                          ; Make a copy
                     08A3  1453 GET_PR_ZNA:                                  ; Continue
58  011C'CF   9E     08A3  1454          MOVAB   ZNABUF,R8                   ; Point to ZNA buffer
    53   58   D0     08A8  1455          MOVL    R8,R3                       ; Make a copy
  83   01   A1     90  08AB  1456          MOVB    1(R1),(R3)+                 ; Enter object type
    50   81   33     08AF  1457 5$:       CVTWB   (R1)+,R0                    ; Get format type, skip over object type
         07   12     08B2  1458          BNEQ    20$                         ; If NEQ then not numbered object
      FF   A1   95   08B4  1459 10$:      TSTB    -1(R1)                      ; Is object type zero ?
         1D   13     08B7  1460          BEQL    40$                         ; If EQL then error
         28   11     08B9  1461          BRB     60$                         ; Else we're done
      FF   A1   95   08BB  1462 20$:      TSTB    -1(R1)                      ; Is object type zero ?
         16   12     08BE  1463          BNEQ    40$                         ; If NEQ then error
    50   01   91     08C0  1464          CMPB    #1,R0                       ; Format type 1 is a counted string
         07   13     08C3  1465          BEQL    30$                         ; If EQL then go move the string
    50   02   91     08C5  1466          CMPB    #2,R0                       ; Format type 2 is UIC + counted string
         0C   12     08C8  1467          BNEQ    40$                         ; If NEQ then format type is unknown
         81   D5     08CA  1468          TSTL    (R1)+                       ; Skip over UIC
    50   81   9A     08CC  1469 30$:      MOVZBL  (R1)+,R0                    ; Get taskname string size
         05   13     08CF  1470          BEQL    40$                         ; If EQL then illegal format
    0C   50   91     08D1  1471          CMPB    R0,#MAX_TASKNAM             ; Is it within bounds?
         09   1B     08D4  1472          BLEQU   50$                         ; If LEQU then legal format
         50   D4     08D6  1473 40$:      CLRL    R0                          ; Else, indicate error
         05   3C     08D8  1474          MOVZWL  #NET$C_DR_FMT,-             ; Setup network failure code
0008'CF             08DA  1475                      NET_L_REASON(R5)            ;
         08   11     08DD  1476          BRB     70$                         ; Take common exit
63  61   50   28     08DF  1477 50$:      MOVC3   R0,(R1),(R3)                ; Enter string
57  53   58   C3     08E3  1478 60$:      SUBL3   R8,R3,R7                    ; Get string size
    50   01   D0     08E7  1479          MOVL    #1,R0                       ; Indicate success
              05     08EA  1480 70$:      RSB
                     08EB  1481
                     08EB  1482                      .DSABL  LSB
```

```
                     08EB  1484 .SBTTL  TELL_DRV        - Call NETDRIVER
                     08EB  1485 ;++
                     08EB  1486 ;
                     08EB  1487 ; Call NETDRIVER to perform a given function.
                     08EB  1488 ;
                     08EB  1489 ; Inputs:
                     08EB  1490 ;
                     08EB  1491 ;       NET_L_R0-R5 = Arguments to NETDRIVER function
                     08EB  1492 ;--
                     08EB  1493 TELL_DRV:                                    ; Tell driver about process
50   0000'CF    7D   08EB  1494          MOVQ    NET_L_R0,R0                ; Get regs for call
52   0008'CF    7D   08F0  1495          MOVQ    NET_L_R2,R2                ;
54   0010'CF    7D   08F5  1496          MOVQ    NET_L_R4,R4                ;
        F703'   30   08FA  1497          BSBW    CALL_NETDRIVER             ; Call driver
               05    08FD  1498          RSB
```

```
                                08FE  1500 .SBTTL  UP_CASE - Upcase the LOGINOUT strings
                                08FE  1501 ;+
                                08FE  1502 ;
                                08FE  1503 ; The NCB (up to the "/"), the access control strings, the taskname, and the
                                08FE  1504 ; process name are up-cased in place.
                                08FE  1505 ;
                                08FE  1506 ; INPUTS:        none
                                08FE  1507 ;
                                08FE  1508 ; OUTPUTS:       none
                                08FE  1509 ;
                                08FE  1510 ;                All register contents are preserved.
                                08FE  1511 ;
                                08FE  1512 ;-
                                08FE  1513 UP_CASE:                                     ; Up-case strings passed to LOGINOUT
                      3F   BB   08FE  1514          PUSHR   #^M<R0,R1,R2,R3,R4,R5>      ; Save regs
                55  0000'CF   9E   0900  1515          MOVAB   NET$AB_UPASCNUM,R5          ; Get translation table
                    54   2F   90   0905  1516          MOVB    #^A'/',R4                   ; Setup terminator
                53  0024'CF   9E   0908  1517          MOVAB   NET_Q_NCB,R3                ; Point to NCB descriptor
                      3C   10   090D  1518          BSBB    UP_IT                       ; Up-case it in place
                      54   D4   090F  1519          CLRL    R4                          ; Say "no terminator"
                53  0034'CF   9E   0911  1520          MOVAB   NET_Q_TSK,R3                ; Point to task-name descriptor
                      33   10   0916  1521          BSBB    UP_IT                       ; Up-case it in place
                53  002C'CF   9E   0918  1522          MOVAB   NET_Q_PRC,R3                ; Point to process-name descriptor
                      2C   10   091D  1523          BSBB    UP_IT                       ; Up-case it in place
                53  003C'CF   9E   091F  1524          MOVAB   NET_Q_ACC,R3                ; Get access control descriptor
              51    04   A3   D0   0924  1525          MOVL    4(R3),R1                    ; Get pointer to strings
                    51   02   C0   0928  1526          ADDL    #2,R1                       ; Skip over flags word
                    52   81   9A   092B  1527          MOVZBL  (R1)+,R2                    ; Get count of bytes in username
                         36   10   092E  1528          BSBB    UP_CASE_LOOP                ; Start at end of loop
                    52   81   9A   0930  1529          MOVZBL  (R1)+,R2                    ; Get count of bytes in password
                         31   10   0933  1530          BSBB    UP_CASE_LOOP                ; Start at end of loop
                    52   81   9A   0935  1531          MOVZBL  (R1)+,R2                    ; Get count of bytes in account name
                         2C   10   0938  1532          BSBB    UP_CASE_LOOP                ; Start at end of loop
            7E  04   A3   63   C1   093A  1533          ADDL3   (R3),4(R3),-(SP)            ; Get address of end of strings
                8E   51   D1   093F  1534          CMPL    R1,(SP)+                    ; Have we gone too far?
                         03   1A   0942  1535          BGTRU   10$                         ; If GTRU then yes
                         3F   BA   0944  1536          POPR    #^M<R0,R1,R2,R3,R4,R5>      ; Restore regs
                              05   0946  1537          RSB
                                   0947  1538
                                   0947  1539
                                   0947  1540 10$:     BUG_CHECK  NETNOSTATE,FATAL         ; Access control strings setup
                                   094B  1541                                             ; incorrectly
                                   094B  1542
                                   094B  1543
                                   094B  1544                 .ENABL  LSB
                                   094B  1545
                    52   63   3C   094B  1546 UP_IT:    MOVZWL  (R3),R2                     ; Get string length
                  51    04   A3   D0   094E  1547          MOVL    4(R3),R1                    ; Point to string
                         12   11   0952  1548          BRB     UP_CASE_LOOP                ; Start at end of loop
                    50   81   90   0954  1549 20$:     MOVB    (R1)+,R0                    ; Get next character
                    54   50   91   0957  1550          CMPB    R0,R4                       ; Is it the terminator?
                         0D   13   095A  1551          BEQL    60$                         ; If EQL yes, we're done
                 50    6540   90   095C  1552          MOVB    (R5)[R0],R0                 ; Up-case it
                         04   13   0960  1553          BEQL    UP_CASE_LOOP                ; If EQL then not alpha-numeric
                FF   A1   50   90   0962  1554          MOVB    R0,-1(R1)                   ; Store up-cased value
                                   0966  1555
                                   0966  1556 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
ACC                     = 0000000C
ACC$K_TERMLEN           = 00000054
ACP$C_STA_F             = 00000004
ACP$C_STA_H             = 00000005
ACP$C_STA_I             = 00000000
ACP$C_STA_N             = 00000001
ACP$C_STA_R             = 00000002
ACP$C_STA_S             = 00000003
BBUF                      0000010E R      02
BIT...                  = 00000006
BUG$_ACPMBFAIL            ********   X    04
BUG$_NETNOSTATE           ********   X    05
BUILD_NCB                 000004BC R      04
CALL_NETDRIVER            ********   X    04
CNF                     = 00000024
CNF$CLR_FIELD             ********   X    04
CNF$COPY                  ********   X    04
CNF$C_LENGTH            = 00000024
CNF$DELETE                ********   X    05
CNF$GET_FIELD             ********   X    04
CNF$INIT_UTL              ********   X    04
CNF$INSERT                ********   X    04
CNF$KEY_SEARCH            ********   X    05
CNF$PURGE                 ********   X    05
CNF$PUT_FIELD             ********   X    05
CNF$_ADVANCE            = 00000000
CNF$_QUIT               = 00000002
CNF$_TAKE_CURR          = 00000003
CNF$_TAKE_PREV          = 00000001
COM$POST                  ********   X    04
CONNECT                   000001D7 R      04
CONN_SPACE              = 000003E8
CREATE_LLI                00000000 R      04
CREATE_SPI                00000834 R      04
DEAL_XWB                  00000090 R      05
DELPROC                   000001E2 R      04
DET_AB_ACC                00000044 R      02
DET_C_ACC               = 00000005
DLE$PRC_EXIT              ********   X    04
ENDBUF                    0000011C R      02
EXE$EPID_TC_IPID          ********   X    04
EXIT_BUF                  00000018 R      03
EXIT_CODE                 0000005C R      02
EXIT_ID                   00000058 R      02
EXIT_MSG                  00000058 R      02
GET_PROC                  0000055A R      04
GET_PR_NAM                00000899 R      04
GET_PR_ZNA                000008A3 R      04
INT_B_PRX                 0000004A R      02
IO$_READVBLK              ********   X    04
IRP$L_IOST1             = 00000038
IRP$L_IOST2             = 0000003C
IRP$L_UCB               = 0000001C
LGI$_INVPWD               ********   X    04
LLI$Z_NDC_LZ            = 00000024
LLI$Z_NDC_RT            = 0C000008
LSB                     = 00000000
LSB$B_R_CXBCNT          = 00000028
LSB$B_R_CXBQUO          = 00000029
LSB$B_SPARE             = 0000002A
LSB$B_STS               = 0000002B
LSB$B_X_ADJ             = 0000000B
LSB$B_X_CXBACT          = 0000000D
LSB$B_X_CXBCNT          = 0000000F
LSB$B_X_CXBQUO          = 0000000E
LSB$B_X_PKTWND          = 0000000C
LSB$B_X_REQ             = 0000000A
LSB$L_CROSS             = 0000002C
LSB$L_R_CXB             = 00000020
LSB$L_R_IRP             = 0000001C
LSB$L_X_CXB             = 00000018
LSB$L_X_IRP             = 00000014
LSB$L_X_PND             = 00000010
LSB$M_BOM               = 00000020
LSB$M_EOM               = 00000040
LSB$M_LI                = 00000001
LSB$S_LSB               = 00000030
LSB$S_SPARE             = 00000004
LSB$S_STS               = 00000001
LSB$V_BOM               = 00000005
LSB$V_EOM               = 00000006
LSB$V_LI                = 00000000
LSB$V_SPARE             = 00000001
LSB$W_HAA               = 00000008
LSB$W_HAR               = 00000006
LSB$W_HAX               = 00000026
LSB$W_HNR               = 00000024
LSB$W_HXS               = 00000004
LSB$W_LNX               = 00000002
LSB$W_LUX               = 00000000
LTB$L_SLOTS             = 00000010
LTB$L_SLT_NXT           = 00000000
LTB$L_XWB               = 0000000C
MAX_TASKNAM             = 0000000C
MBX_ACTION                000000E6 R      04
MBX_CHAN                  0000004C R      02
MBX_IOSB                  00000050 R      02
MBX_LEN                   00000052 R      02
MBX_MSG_LTH             = 00000096
MBX_PID                   00000054 R      02
MBX_RDCNT                 0000004E R      02
MBX_UNIT                  0000011A R      02
MSG$_CONNECT            = 00000032
MSG$_DELPROC            = 00000003
MSG$_PATHLOST           = 00000036
MSG$_RESET              = 00000041
NCB_DATA                  0000005C R      02
NDC$C_LENGTH            = 0000001C
NET$AB_UPASCNUM           ********   X    04
NET$ACQUIRE_NDCOU         ********   X    04
NET$ALLOCATE              ********   X    04
NET$ALONPAGED             ********   X    04
NET$BIN2ASC               ********   X    04
```

```
NET$CONNECT_FAIL                  0000024C RG    04
NET$CREATE_MBX                    0000004C RG    04
NET$C_ACT_TIMER                 = 0000001E
NET$C_DR_ACCESS                 = 00000022
NET$C_DR_EXIT                   = 00000026
NET$C_DR_FMT                    = 00000005
NET$C_DR_IMLONG                 = 0000002B
NET$C_DR_NOBJ                   = 00000004
NET$C_DR_RSU                    = 00000001
NET$C_DYN_WQE                     ********    X  02
NET$C_EFN_ASYN                  = 00000002
NET$C_EFN_WAIT                  = 00000001
NET$C_IPL                       = 00000008
NET$C_MAXACCFLD                 = 00000027
NET$C_MAXLINNAM                 = 0000000F
NET$C_MAXLNK                    = 000003FF
NET$C_MAXNODNAM                 = 00000006
NET$C_MAXOBJNAM                 = 0000000C
NET$C_MAX_AREAS                 = 0000003F
NET$C_MAX_LINES                 = 00000040
NET$C_MAX_NCB                   = 0000006E
NET$C_MAX_NODES                 = 000003FF
NET$C_MAX_OBJ                   = 000000FF
NET$C_MAX_WQE                   = 00000014
NET$C_MINBUFSIZ                 = 000000C0
NET$C_TID_ACT                   = 00000003
NET$C_TID_RUS                   = 00000001
NET$C_TID_XRT                   = 00000002
NET$C_TRCTL_CEL                 = 00000002
NET$C_TRCTL_OVR                 = 00000005
NET$C_UTLBUFSIZ                 = 00001000
NET$DEALLOCATE                    ********    X  05
NET$DECR_MCOUNT                   ********    X  05
NET$DEC_TRANS                     ********    X  04
NET$DELIVER_CI                    000003DC R      04
NET$DLL_X25_CALL                  ********    X  04
NET$DLL_X25_RESET                 ********    X  04
NET$DRV_CANCEL                    ********    X  04
NET$FLUSH_LLI_CNT                 ********    X  05
NET$GETUTLBUF                     ********    X  04
NET$GL_CNR_LLI                    ********    X  05
NET$GL_CNR_NDI                    ********    X  04
NET$GL_CNR_OBI                    ********    X  04
NET$GL_CNR_SPI                    ********    X  04
NET$GL_NET_UCB                    ********    X  04
NET$GL_PTR_VCB                    ********    X  04
NET$GQ_MBX_NAME                   00000020 RG    03
NET$GQ_WQE_MBX                    000000F6 RG    02
NET$KILL_MBX                      0000008B RG    04
NET$MBX_AST                       000000EE RG    04
NET$MBX_QIO                       00000098 RG    04
NET$M_MAXLNKMSK                 = 000003FF
NET$PROC_XWB                      00000000 RG    05
NET$RELEASE_NDCOU                 ********      X  05
NET$RESEND_SERVER                 000002CA RG    04
NET$SCAN_FOR_ZNA                  00000287 RG    04
NET$SERVER_FAIL                   0000026D RG    04
NET$SET_MBX_AST                   000000C9 RG    04
NET$STARTUP_OBJ                   00000317 RG    04
NET$STARTUP_OBJ_NAM               000002FD RG    04
NETUPD$_ABORT                   = 00000001
NETUPD$_CONNECT                 = 00000002
NETUPD$_EXIT                    = 00C00C03
NETUPD$_PROCRE                  = 00000004
NET_A_LCI                         00000020 R     02
NET_A_NCB                         00000010 R     02
NET_L_FCT                         00000000 R     02
NET_L_LNK                         0000000C R     02
NET_L_LPD                         00000004 R     02
NET_L_PID                         00000004 R     02
NET_L_R0                          00000000 R     02
NET_L_R1                          00000004 R     02
NET_L_R2                          00000008 R     02
NET_L_R3                          0000000C R     02
NET_L_R4                          00000010 R     02
NET_L_R5                          00000014 R     02
NET_L_REASON                      00000008 R     02
NET_L_UCB                         00000014 R     02
NET_Q_ACC                         0000003C R     02
NET_Q_IMAGE                       00000045 R     03
NET_Q_NCB                         00000024 R     02
NET_Q_NETPREFIX                   00000000 R     03
NET_Q_PRC                         0000002C R     02
NET_Q_PROC                        0000005B R     03
NET_Q_SYSTEM                      00000032 R     03
NET_Q_TASKZNA                     0000000B R     03
NET_Q_TSK                         00000034 R     02
NEW_LINK                          0000009E R     05
NFB$C_LLI_XWB                   = 08010017
NFB$C_NDI_NNA                   = 02020043
NFB$C_NDI_TAD                   = 02010010
NFB$C_OBI_IAC                   = 03020043
NFB$C_OBI_NAM                   = 03020044
NFB$C_OBI_NUM                   = 03010014
NFB$C_OBI_PID                   = 03010015
NFB$C_OBI_PRX                   = 03010016
NFB$C_OBI_SFI                   = 03020042
NFB$C_OBI_UCB                   = 03010012
NFB$C_OBI_ZNA                   = 03020041
NFB$C_OP_EQL                    = 00000000
NFB$C_OP_NEQ                    = 00000003
NFB$C_SPI_ACS                   = 12020041
NFB$C_SPI_IRP                   = 12010011
NFB$C_SPI_NCB                   = 12020044
NFB$C_SPI_PID                   = 12010010
NFB$C_SPI_PNM                   = 12020045
NFB$C_SPI_PRL                   = 12000002
NFB$C_SPI_RID                   = 12020042
NFB$C_SPI_RNA                   = 12010013
NFB$C_SPI_SFI                   = 12020043
NMA$C_ACES_BOTH                 = 00000003
NMA$C_ACES_NONE                 = 00000000
NMA$C_ACES_OUTG                 = 00000002
NSP$C_EXT_LNK                   = 0000001E
```

```
NSP$C_MAXHDR                    =  00000009
OBI_B_PRX                          00000049 R      02
PR$_IPL                            ********   X    05
PRC                             =  000000C8
PTR_CON_BUF                        0000001C R      02
PTR_NCB_BUF                        00000018 R      02
RCB$B_ECL_DPX                   =  00000067
RCB$L_PTR_LTB                   =  00000024
RCB$W_CNT_XRE                   =  0000009C
RMS$_FNF                           ********   X    04
SEND_TO_SERVER                     000007B4 R      04
SIZ...                          =  00000001
SS$_ABORT                          ********   X    04
SS$_CANCEL                         ********   X    04
SS$_CONNECFAIL                     ********   X    05
SS$_NORMAL                         ********   X    04
STS$V_INHIB_MSG                 =  0000001C
STS_M_NETLOG                    =  00000080
STS_M_NOACNT                    =  00000008
STS_M_NOAUTH                    =  00000040
SYS$CREMBX                         ********  GX    04
SYS$CREPRC                         ********  GX    04
SYS$DASSGN                         ********  GX    04
SYS$GB_DEFPRI                      ********   X    04
SYS$GETCHN                         ********  GX    04
SYS$QIO                            ********  GX    04
TASKZNA                            00000013 R      03
TELL_DRV                           000008EB R      04
TR$C_MAXHDR                     =  0000001C
TR$C_NI_ALLEND1                 =  040000AB
TR$C_NI_ALLEND2                 =  00000000
TR$C_NI_ALLROU1                 =  030000AB
TR$C_NI_ALLROU2                 =  00000000
TR$C_NI_PREFIX                  =  000400AA
TR$C_NI_PROT                    =  00000360
TR$C_PRI_ECL                    =  0000001F
TR$C_PRI_RTHRU                  =  0000001F
TSK                             =  0000012C
UP_CASE                            000008FE R      04
UP_CASE_LOOP                       00000966 R      04
UP_IT                              0000094B R      04
WQE$C_SUB_MBX                   =  00000005
WQE$INSQUE                         ********   X    04
WQE$L_PM1                       =  00000010
WQE$L_PM2                       =  00000014
WQE_MBX_LTH                     =  00000018
X25_DEV_NAME                       00000077 R      03
XWB                             =  00000000
XWB$B_ACCESS                    =  0000000B
XWB$B_DATA                      =  0000005B
XWB$B_FIPL                      =  0000001F
XWB$B_LOGIN                     =  000000C0
XWB$B_LPRNAM                    =  000000A4
XWB$B_PRO                       =  0000005A
XWB$B_RID                       =  0000006F
XWB$B_RPRNAM                    =  000000B8
XWB$B_SP3                       =  0000006E
XWB$B_STA                       =  0000001E
XWB$B_TYPE                      =  0000000A
XWB$B_X_FLW                     =  0000006C
XWB$B_X_FLWCNT                  =  0000006D
XWB$C_COMLNG                    =  000000A4
XWB$C_CONLNG                    =  00000112
XWB$C_DATA                      =  00000010
XWB$C_LOGIN                     =  00000040
XWB$C_LPRNAM                    =  00000014
XWB$C_NDC_LNG                   =  00000020
XWB$C_NUMSTA                    =  00000008
XWB$C_RID                       =  00000010
XWB$C_RPRNAM                    =  00000014
XWB$C_STA_CAR                   =  00000002
XWB$C_STA_CCS                   =  00000004
XWB$C_STA_CIR                   =  00000003
XWB$C_STA_CIS                   =  00000001
XWB$C_STA_CLO                   =  00000000
XWB$C_STA_DIR                   =  00000006
XWB$C_STA_DIS                   =  00000007
XWB$C_STA_RUN                   =  00000005
XWB$L_DEA_IRP                   =  00000104
XWB$L_FPC                       =  00000020
XWB$L_FR3                       =  00000024
XWB$L_FR4                       =  00000028
XWB$L_ICB                       =  0000010C
XWB$L_IRP_ACC                   =  00000080
XWB$L_LINK                      =  0000002C
XWB$L_ORGUCB                    =  00000010
XWB$L_PID                       =  00000034
XWB$L_VCB                       =  00000030
XWB$L_WLBL                      =  00000004
XWB$L_WLFL                      =  00000000
XWB$M_FLG_BREAK                 =  00000001
XWB$M_FLG_CLO                   =  00000200
XWB$M_FLG_IAVL                  =  00001000
XWB$M_FLG_SCD                   =  00000100
XWB$M_FLG_SDACK                 =  00000008
XWB$M_FLG_SDFL                  =  00004000
XWB$M_FLG_SDT                   =  00000080
XWB$M_FLG_SIACK                 =  00000004
XWB$M_FLG_SIFL                  =  00002000
XWB$M_FLG_SLI                   =  00000010
XWB$M_FLG_TBPR                  =  00000800
XWB$M_FLG_WBP                   =  00000040
XWB$M_FLG_WBUF                  =  00000002
XWB$M_FLG_WDAT                  =  00000400
XWB$M_FLG_WHGL                  =  00000020
XWB$M_PRO_CCA                   =  00000008
XWB$M_PRO_NAR                   =  00000010
XWB$M_PRO_NFC                   =  00000001
XWB$M_PRO_PH2                   =  00000004
XWB$M_PRO_SFC                   =  00000002
XWB$M_STS_ASTPND                =  00000400
XWB$M_STS_ASTREQ                =  00000800
XWB$M_STS_CON                   =  00000010
XWB$M_STS_DIS                   =  00000008
```

```
XWB$M_STS_DTNAK                 = 00000100
XWB$M_STS_LINAK                 = 00000200
XWB$M_STS_NDC                   = 00001000
XWB$M_STS_OVF                   = 00000080
XWB$M_STS_RBP                   = 00000040
XWB$M_STS_SOL                   = 00000004
XWB$M_STS_TID                   = 00000001
XWB$M_STS_TLI                   = 00000002
XWB$M_STS_TMO                   = 00000020
XWB$Q_FORK                      = 00000014
XWB$Q_FREE_CXB                  = 00000118
XWB$R_CON_BLK                   = 000000A4
XWB$R_RUN_BLK                   = 0000C0A4
XWB$S_                          = 00000006
XWB$S_COMLNG                    = 0000006E
XWB$S_CON_BLK                   = 0000006E
XWB$S_DATA                      = 00000010
XWB$S_DT                        = 00000030
XWB$S_FLG                       = 00000002
XWB$S_FORK                      = 00000008
XWB$S_FREE_CXB                  = 00000008
XWB$S_LI                        = 00000030
XWB$S_LOGIN                     = 0000003F
XWB$S_LPRNAM                    = 00000013
XWB$S_NDC                       = 00000020
XWB$S_PRO                       = 00000001
XWB$S_RID                       = 00000010
XWB$S_RPRNAM                    = 00000013
XWB$S_RUN_BLK                   = 00000064
XWB$S_STS                       = 00000002
XWB$S_XWB                       = 00000120
XWB$T_                          = 00000112
XWB$T_DATA                      = 0000005C
XWB$T_DT                        = 000000A4
XWB$T_LI                        = 000000D4
XWB$T_LOGIN                     = 000000CD
XWB$T_LPRNAM                    = 000000A5
XWB$T_RID                       = 00000070
XWB$T_RPRNAM                    = 000000B9
XWB$V_FLG_BREAK                 = 00000000
XWB$V_FLG_CLO                   = 00000009
XWB$V_FLG_IAVL                  = 0000000C
XWB$V_FLG_SCD                   = 00000008
XWB$V_FLG_SDACK                 = 00000003
XWB$V_FLG_SDFL                  = 0000000E
XWB$V_FLG_SDT                   = 00000007
XWB$V_FLG_SIACK                 = 00000002
XWB$V_FLG_SIFL                  = 0000000D
XWB$V_FLG_SLI                   = 00000004
XWB$V_FLG_TBPR                  = 0000000B
XWB$V_FLG_WBP                   = 00000006
XWB$V_FLG_WBUF                  = 00000001
XWB$V_FLG_WDAT                  = 0000000A
XWB$V_FLG_WHGL                  = 00000005
XWB$V_PRO_CCA                   = 00000003
XWB$V_PRO_NAR                   = 00000004
XWB$V_PRO_NFC                   = 00000000
XWB$V_PRO_PH2                   = 00000002
XWB$V_PRO_SFC                   = 00000001
XWB$V_STS_ASTPND                = 0000000A
XWB$V_STS_ASTREQ                = 0000000B
XWB$V_STS_CON                   = 00000004
XWB$V_STS_DIS                   = 00000003
XWB$V_STS_DTNAK                 = 00000008
XWB$V_STS_LINAK                 = 00000009
XWB$V_STS_NDC                   = 0000000C
XWB$V_STS_OVF                   = 00000007
XWB$V_STS_RBP                   = 00000006
XWB$V_STS_SOL                   = 00000002
XWB$V_STS_TID                   = 00000000
XWB$V_STS_TLI                   = 00000001
XWB$V_STS_TMO                   = 00000005
XWB$W_CI_PATH                   = 00000110
XWB$W_DECAY                     = 0000004E
XWB$W_DLY_FACT                  = 00000056
XWB$W_DLY_WGHT                  = 00000058
XWB$W_ELAPSE                    = 0000004A
XWB$W_FLG                       = 0000001C
XWB$W_LOCLNK                    = 0000003E
XWB$W_LOCSIZ                    = 00000040
XWB$W_PATH                      = 00000038
XWB$W_PROGRESS                  = 00000052
XWB$W_REFCNT                    = 0000000C
XWB$W_REMLNK                    = 0000003C
XWB$W_REMNOD                    = 0000003A
XWB$W_REMSIZ                    = 00000042
XWB$W_RETRAN                    = 00000054
XWB$W_R_REASON                  = 00000044
XWB$W_SIZE                      = 00000008
XWB$W_STS                       = 0000000E
XWB$W_TIMER                     = 00000050
XWB$W_TIM_ID                    = 00000048
XWB$W_TIM_INACT                 = 0000004C
XWB$W_X_REASON                  = 00000046
XWB$Z_NDC                       = 00000084
ZNABUF                            0000011C R      02
_$$_                            = 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
_255DUA28:[SHRLIB]NMALIBRY.MLB;1	1
_255DUA28:[SHRLIB]EVCDEF.MLB;1	0
_255DUA28:[NETACP.OBJ]NETDRV.MLB;1	1
_255DUA28:[NETACP.OBJ]NET.MLB;1	17
_255DUA28:[SYS.OBJ]LIB.MLB;1	6
_255DUA28:[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$:NETDRV/LIB+SHRLIB$
```

NETDRVXPT
LIS

NETOPCOM
LIS

NETLLCNT
LIS

NETEVTLOG
LIS

NETPROCRE
LIS

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