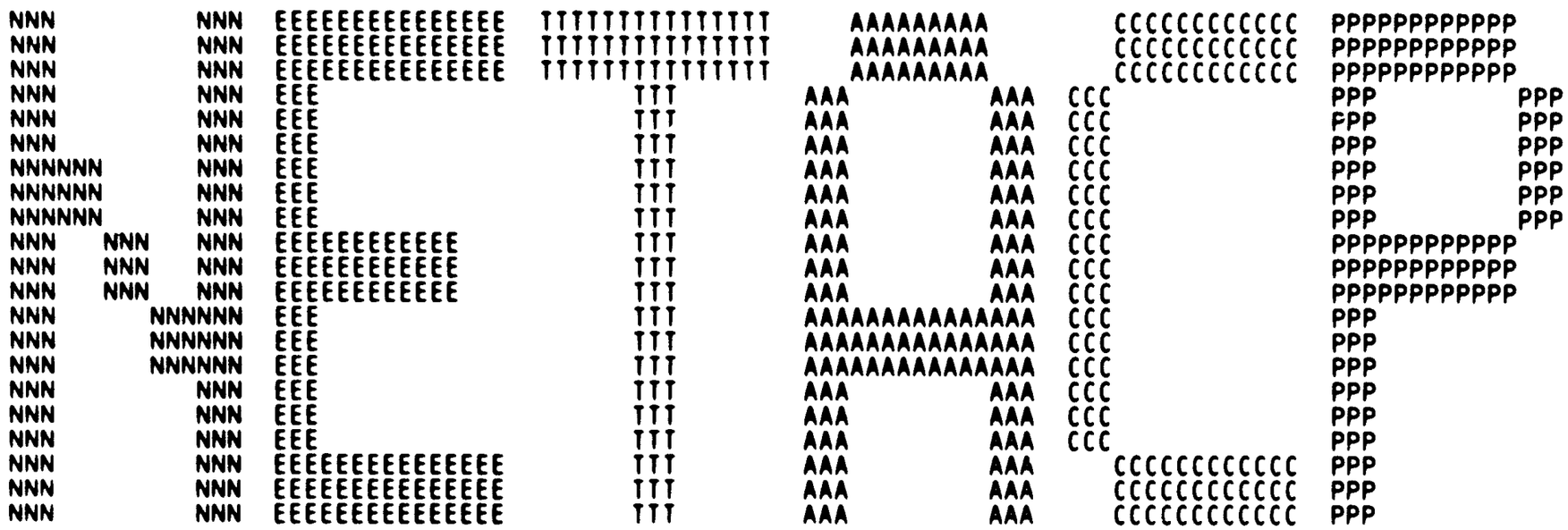
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN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

```
NN       NN  EEEEEEEEEE  TTTTTTTTTT    CCCCCCCC  TTTTTTTTTT  LL            AAAAAA    LL          LL
NN       NN  EEEEEEEEEE  TTTTTTTTTT    CCCCCCCC  TTTTTTTTTT  LL            AAAAAA    LL          LL
NN       NN  EE              TT      CC              TT      LL          AA      AA  LL          LL
NN       NN  EE              TT      CC              TT      LL          AA      AA  LL          LL
NNNN     NN  EE              TT      CC              TT      LL          AA      AA  LL          LL
NNNN     NN  EE              TT      CC              TT      LL          AA      AA  LL          LL
NN   NN  NN  EEEEEEEE        TT      CC              TT      LL          AA      AA  LL          LL
NN   NN  NN  EEEEEEEE        TT      CC              TT      LL          AA      AA  LL          LL
NN     NNNN  EE              TT      CC              TT      LL          AAAAAAAAAA  LL          LL
NN     NNNN  EE              TT      CC              TT      LL          AAAAAAAAAA  LL          LL
NN       NN  EE              TT      CC              TT      LL          AA      AA  LL          LL              ....
NN       NN  EE              TT      CC              TT      LL          AA      AA  LL          LL              ....
NN       NN  EEEEEEEEEE      TT        CCCCCCCC      TT      LLLLLLLLLL  AA      AA  LLLLLLLLLL  LLLLLLLLLL      ....
NN       NN  EEEEEEEEEE      TT        CCCCCCCC      TT      LLLLLLLLLL  AA      AA  LLLLLLLLLL  LLLLLLLLLL      ....

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

Table of contents

```
0000     1          .TITLE  NETCTLALL - Process ACP control Qio's
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  ; FACILITY:     NETWORK ACP
0000    29  ;
0000    30  ; ABSTRACT:     This module processes control QIO's to NETACP.
0000    31  ;
0000    32  ; ENVIRONMENT:  MODE = KERNEL
0000    33  ;
0000    34  ; AUTHOR:       A.ELDRIDGE, CREATION DATE: 8-JAN-80
0000    35  ;
0000    36  ; MODIFIED BY:
0000    37  ;
0000    38  ;       V03-023 PRB0341         Paul Beck       20-Jul-1984  18:35
0000    39  ;               Fix problem whereby the returned P2 parameter for SHOW
0000    40  ;               functions could be occasionally garbaged.
0000    41  ;
0000    42  ;       V022    PRB0332         Paul Beck        1-MAY-1984 20:25
0000    43  ;               Store EPID instead of IPID in OBI$L_PID.
0000    44  ;
0000    45  ;       V021    RNG0021         Rod Gamache     07-Feb-1984
0000    46  ;               Fix crash that resulted from internal pool allocation failure
0000    47  ;               with an invalid string length returned, that was attempted to
0000    48  ;               be copied on the stack (which got an INVALID STACK error)!
0000    49  ;               Fix size return of P4 buffer to not return half filled
0000    50  ;               parameter data.
0000    51  ;
0000    52  ; Previous modifications by:
0000    53  ;
0000    54  ;       A.Eldridge,S.Davis,T.Halvorsen,R.Gamache
```

```
                     56          .SBTTL  DECLARATIONS
                     57 ;
                     58 ; INCLUDE FILES:
                     59 ;
                     60          $ABDDEF
                     61          $IRPDEF
                     62          $UCBDEF
                     63          $PRVDEF
                     64
                     65          $NETSYMDEF
                     66          $NETUPDDEF
                     67
                     68          $DRDEF
                     69          $CNFDEF
                     70          $CNRDEF
                     71          $NFBDEF
                     72          $RCBDEF
                     73
                     74
                     75 ;
                     76 ; OWN STORAGE:
                     77 ;
                     78          .PSECT  NET_IMPURE,WRT,NOEXE,LONG
                     79
                     80 ;
                     81 ; Define storage for control QIO processing
                     82 ;
00000004 0000        83 NET$GL_PM_OUT:          .BLKL   1       ; Value returned as the NFB 'parameter'
00000008 0004        84 NET$GL_PM_IN:           .BLKL   1       ; Value supplied as the NFB 'parameter'
                     85
                     86 ;
                     87 ; Define the search key list to be used to re-establish the position
                     88 ; in the database from the NFB context.  The list here contains exactly
                     89 ; two entries (the primary and secondary keys).  A key which isn't
                     90 ; desired is indicated by having a field ID of NFB$C_WILDCARD.
                     91 ;
                     92
                     93 NET$AL_SRCH_LIST:
                     94
0 0008               95 NET$GL_SRCH_ID::        .BLKL   1       ; QIO "search" key field i.d.
0 000C               96 NET$GL_OPER:            .BLKL   1       ; Type of comparison for primary key
00000018 0010        97 NET$GQ_SRCH_KEY::       .BLKL   2       ; Value/descriptor of the "search" key
                     98
0000001C 0018        99 NET$GL_SRCH2_ID::       .BLKL   1       ; Secondary search key field ID
00000020 001C       100 NET$GL_OPER2:           .BLKL   1       ; Type of comparison for secondary key
00000028 0020       101 NET$GQ_SRCH2_KEY::      .BLKL   2       ; Value of secondary search key
                    102
00000000 0028       103                         .LONG   0       ; Terminate list
                    104
                    105 ;*****
                    106 ;
                    107 ;  The following 8 longwords must be together, in order.  The desciptors
                    108 ;  are used to hold the original IO$_ACPCONTROL buffer descriptors.  They
                    109 ;  are also used as the descriptors of the buffers used for the re-issuing
                    110 ;  of the control QIOs to the X.25 ACP.
                    111 ;
                    112 ;*****
```

```
                     002C   113
00000030             002C   114 NET$GL_SIZ_P4::         .BLKL   1       ; Length of result buffer
00000034             0030   115 NET$GL_PTR_P4::         .BLKL   1       ; Pointer to result buffer
00000038             0034   116 NET$GL_SIZ_P3::         .BLKL   1       ; Length of and pointer to field to rcv
0000003C             0038   117 NET$GL_PTR_P3::         .BLKL   1       ; # of bytes returned P4 buffer
00000040             003C   118 NET$GL_SIZ_P2::         .BLKL   1       ; Length of input string
00000044             0040   119 NET$GL_PTR_P2::         .BLKL   1       ; Pointer to input string
00000048             0044   120 NET$GL_SIZ_P1::         .BLKL   1       ; Length of Net Function Block
0000004C             0048   121 NET$GL_PTR_P1::         .BLKL   1       ; Pointer to Net Function Block
                     004C   122
000000C8             004C   123 DUMMY_P2_LNG  =  200
000000C8             004C   124 DUMMY_P4_LNG  =  200
                     004C   125
                     004C   126 DUMMY_P4:                               ; Shared dummy P2/P4 buffer in case
00000114             004C   127 DUMMY_P2:          .BLKB   DUMMY_P4_LNG ; either was optional and not supplied
00000000             0114   128 DUMMY_P3:          .LONG   0            ; Dummy P3 buffer in case none supplied
                     0118   129
00000000             0118   130 SIZ_L_P4:          .LONG   0            ; Local P4 buffer size field
00000000             011C   131 PTR_L_P4:          .LONG   0            ; Local P4 buffer pointer
00000000             0120   132 PTR_L_OLDP4:       .LONG   0            ; Local old P4 buffer pointer
                     0124   133
00000000             0124   134 PTR_CNFCNT:        .LONG   0            ; Pointer to count of CNFs processed
00000000             0128   135 PTR_OLD_CNF:       .LONG   0            ; Pointer to CNF being replaced
                     012C   136
00000000             012C   137 LOCAL_L_FLAG:      .LONG   0            ; For LOCAL "line" check
00000000             0130   138 P4_ABD_CNT:        .LONG   0            ; Address of P4 ABD count field
00000000             0134   139 P2_ABD_CNT:        .LONG   0            ; Address of P2 ABD count field
00000000             0138   140 P1_ABD_CNT:        .LONG   0            ; Address of P1 ABD count field
00000000             013C   141 GET_W_STATUS:      .LONG   0            ; Storage for CNF$GET_FIELD call status
00000000 00000000    0140   142 QUAD_BUF:         .QUAD   0            ; A scratch buffer
00000000 00000000    0148   143 CTL_Q_DCLZNA:     .QUAD   0            ; Desciptor of the following
00000160             0150   144 CTL_DCLZNA:       .BLKB   NET$C_MAXOBJNAM+4 ; For holding Declared Object number
                     0160   145                                         ; and name plus 3 bytes slop
                     0160   146
00000162             0160   147 NET$GW_X25_CHAN:: .BLKW 1                ; Channel to the X25 ACP
                     0162   148 SPI_CANCEL_SRCH:
                     0162   149                   .CNFFLD spi,l,pid     ; Primary search key field ID
00000000             0166   150                   .LONG   NFB$C_OP_EQL  ; Primary operator
00000000             016A   151                   .LONG   0             ; Quadword primary search value
00000000             016E   152 CANCEL_L_PID:     .LONG   0             ; For holding PID of canceller
                     0172   153                   .CNFFLD spi,l,chn     ; Secondary search key field ID
00000000             0176   154                   .LONG   NFB$C_OP_EQL  ; Secondary operator
00000000             017A   155                   .LONG   0             ; Quadword secondary search value
00000000             017E   156 CANCEL_W_CHN:     .LONG   0             ; For holding channel of canceller
00000000             0182   157                   .LONG   0             ; - End of search list
```

```
                         00000000   159          .PSECT  NET_PURE,NOWRT,NOEXE,LONG
                             0000   160
                             0000   161 ;
                             0000   162 ; Mask identifying all databases maintained exclusively by X.25 ACP
                             0000   163 ;
                             0000   164
                         0BE3FE00    0000   165 X25_DB_MASK:     .LONG   <1@NFB$C_DB_XNI>!-
                                     0004   166                         <1@NFB$C_DB_XDI>!-
                                     0004   167                         <1@NFB$C_DB_XGI>!-
                                     0004   168                         <1@NFB$C_DB_XS5>!-
                                     0004   169                         <1@NFB$C_DB_XD5>!-
                                     0004   170                         <1@NFB$C_DB_XS9>!-
                                     0004   171                         <1@NFB$C_DB_XD9>!-
                                     0004   172                         <1@NFB$C_DB_XTI>!-
                                     0004   173                         <1@NFB$C_DB_XTT>!-
                                     0004   174                         <1@NFB$C_DB_XAI>!-
                                     0004   175                         <1@NFB$C_DB_PSI1>!-
                                     0004   176                         <1@NFB$C_DB_PSI2>!-
                                     0004   177                         <1@NFB$C_DB_PSI3>!-
                                     0004   178                         <1@NFB$C_DB_PSI4>!-
                                     0004   179                         <1@NFB$C_DB_PSI5>
                                     0004   180
      3A 57 4E 5F 0000000C'010E0000' 0004   181 NET$GQ_X25_DEV::         .ASCID "_NW:"   ; X25 device name
                                     0010   182
                                     0010   183
                                     0010   184 ASSUME  PRV$V_DIAGNOSE LE 31             ; Insure bits are in low order
                                     0010   185 ASSUME  PRV$V_OPER     LE 31             ; longword
                                     0010   186
                                     0010   187 .MACRO  NFB_CHAR   FCT,WRTBCK,PRVLIST    ; Define NFB fct characteristics
                                     0010   188         TMPMASK = 0                     ; Init writeback mask
                                     0010   189         .IRP    A,<WRTBCK>
                                     0010   190                 TMPMASK = TMPMASK!<1@'A>
                                     0010   191         .ENDR
                                     0010   192         .=WRTBCK+FCT+NFB$C_'FCT          ; Find writeback cell
                                     0010   193         .BYTE   TMPMASK                 ; Enter writeback mask
                                     0010   194
                                     0010   195         TMPMASK = 0                     ; Note that only the low order
                                     0010   196         .IRP    A,<PRVLIST>             ; longword of the priv mask is used
                                     0010   197                 TMPMASK = TMPMASK!<1@<PRV$V_'A>>
                                     0010   198         .ENDR
                                     0010   199         .=PRV_Q_REQ+<8*NFB$C_'FCT>
                                     0010   200         .LONG   TMPMASK                 ; Setup privilege mask
                                     0010   201 .ENDM
                                     0010   202
00000000'00000000'00000000'00000000' 0010   203 PRV_Q_REQ:  .LONG   0[NFB$C_FC_MAX+1]   ; Required privilege
00000000'00000000'00000000'00000000' 0020
00000000'00000000'00000000'00000000' 0030
00000000'00000000'00000000'00000000' 0040
00000000'00000000'00000000'00000000' 0050
00000000'00000000'00000000'00000000' 0060
00000000'00000000'00000000'00000000' 0070
00000000'00000000'00000000'00000000' 0080
00000000'00000000'00000000'00000000' 0090
         00000000'00000000'00000000' 00A0
00000000'00000000'00000000'00000000' 00AC   204             .LONG   0[NFB$C_FC_MAX+1]   ; masks
00000000'00000000'00000000'00000000' 00BC
00000000'00000000'00000000'00000000' 00CC
```

```
00000000'00000000'00000000'00000000' 00DC
00000000'00000000'00000000'00000000' 00EC
00000000'00000000'00000000'00000000' 00FC
00000000'00000000'00000000'00000000' 010C
00000000'00000000'00000000'00000000' 011C
00000000'00000000'00000000'00000000' 012C
        00000000'00000000'00000000' 013C
00'00'00'00'00'00'00'00'00'00'00'00' 0148   205 WRTBCKFCT:  .BYTE   0[NFB$C_FC_MAX+1]   ; NFB functions requiring write-back
00'00'00'00'00'00'00'00'00'00'00'00' 0154
00'00'00'00'00'00'00'00'00'00'00'00' 0160
                        00'00'00'    016C
                                     016F   206
                                     016F   207
                                     016F   208          .ALIGN  LONG
                                     0170   209
```

```
00000170  0170   211 TMP=.
          0170   212
          0170   213     NFB_CHAR    LOGEVENT,   <>,          <>
          00F4   214     NFB_CHAR    READEVENT,  <1,4>,       <OPER>
          00FC   215
          00FC   216     NFB_CHAR    DECLNAME,   <>,          <SYSNAM>
          00BC   217     NFB_CHAR    DECLOBJ,    <>,          <SYSNAM>
          00C4   218     NFB_CHAR    DECLSERV,   <>,          <>
          00CC   219
          00CC   220     NFB_CHAR    FC_SET,     <2>,         <OPER>
          012C   221     NFB_CHAR    FC_CLEAR,   <2>,         <OPER>
          0134   222     NFB_CHAR    FC_DELETE,  <2>,         <OPER>
          011C   223     NFB_CHAR    FC_SHOW,    <2,4>,       <>
          0124   224     NFB_CHAR    FC_ZERCOU,  <2,4>,       <OPER>
          013C   225
00000170  013C   226 .=TMP
          0170   227
       00000000  228         .PSECT  NET_CODE,NOWRT,LONG,EXE
```

```
                                        .SBTTL  DISPATCHING
                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ; NET$CONTROL_QIO - DETERMINE WHICH CONTROL FUNCTION HAS BEEN
                                ;                  REQUESTED AND DISPATCH TO IT.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       BSB     NET$CONTROL_QIO
                                ;
                                ; INPUT PARAMETERS:
                                ;
                                ;       R3 - IRP address
                                ;       R5 - UCB address
                                ;
                                ;       ACP Control Block - generally has the following args:
                                ;
                                ;               P1 - (FIB) 1 byte of function code, 4 bytes of parameter
                                ;               P2 - Supplies key into data base (counted or uncounted)
                                ;               P3 - Returns result length
                                ;               P4 - Returns result buffer
                                ;
                                ;
                                ; COMPLETION CODES:
                                ;
                                ;       SS$_BADPARAM    Bad or conflicting parameter(s)
                                ;       SS$_DIRFULL     No room in connect name table
                                ;       SS$_INSFMEM     Couldn't allocate a control block
                                ;       SS$_NOMBX       No associated mbx for declared name or object
                                ;       SS$_NOPRIV      No privilege for requested operation
                                ;       SS$_NORMAL      Successful completion
                                ;       SS$_NOSUCHNODE  Unknown node or line
                                ;       SS$_RESULTOVF   Supplied result buffer too short
                                ;       SS$_WRITLCK     Attempt to write a read-only parameter
                                ;       SS$_ILLCNTRFUNC Unrecognized controller function
                                ;
                                ;       OTHER CODES FROM $ASSIGN, $QIO
                                ;
                                ;--
                                NET$CONTROL_QIO::
                                        ;
                                        ;   Set up pointers to all strings in the funny ACP buffer.
                                        ;
      50   2C B3    D0  0000           MOVL    @IRP$L_SVAPTE(R3),R0    ; Get the complex bfr address
           52   04  9A  0004           MOVZBL  #ABD$C_RES,R2           ; Get value of P4 type for loop
   5B   002C'CF    9E  0007           MOVAB   NET$GL_SIZ_P4,R11       ; Get table address for loop
                       000C
                       000C     10$:   ASSUME  ABD$W_TEXT EQ 0
                       000C
56 50   52   08  7A  000C           EMUL    #ABD$C_LENGTH,R2,R0,R6  ; Get address of offset
        7E   66  3C  0011           MOVZWL  (R6),-(SP)              ; Get offset
     8B   02 A6  3C  0014           MOVZWL  ABD$W_COUNT(R6),(R11)+  ; Store the parameter lth
        56   8E  C0  0018           ADDL    (SP)+,R6                ; Get address of text
     8B   01 A6  DE  001B           MOVAL   1(R6),(R11)+            ; Store pointer to text area
                     001F                                           ; (biased for access mode)
         EA 52   F5  001F           SOBGTR  R2,10$                  ; Loop
```

```
                                0022   287          ;
                                0022   288          ;
                                0022   289          ;   Zero the 'window' descriptor in the ABD so that it is not written
                                0022   290          ;   back when the IRP completes.  Also, save pointers to the P1, P2,
                                0022   291          ;   and P4 descriptor count fields so that they may eventually be
                                0022   292          ;   zeroed since these buffers are conditionally written back.
                                0022   293          ;
                     02 A0   B4 0022   294          CLRW    <ABD$C_LENGTH*ABD$C_WINDOW>+ ABD$W_COUNT(R0)
          0138'CF    0A A0   9E 0025   295          MOVAB   <ABD$C_LENGTH*ABD$C_FIB>    + ABD$W_COUNT(R0),P1_ABD_CNT
          0134'CF    12 A0   9E 002B   296          MOVAB   <ABD$C_LENGTH*ABD$C_NAME>   + ABD$W_COUNT(R0),P2_ABD_CNT
          0130'CF    22 A0   9E 0031   297          MOVAB   <ABD$C_LENGTH*ABD$C_RES>    + ABD$W_COUNT(R0),P4_ABD_CNT
                                0037   298
                                0037   299          ;
                                0037   300          ;   Initialize miscellaneous info used by action routines
                                0037   301          ;
                    0000'CF     7C 0037   302          CLRQ    NET$GQ_USR_STAT         ; Init user's IOSB image
                    0000'CF     D4 003B   303          CLRL    NET$GL_PM_OUT           ; Init NFB output parameter
                                003F   304          ;
                                003F   305          ;
                                003F   306          ;   Verify that the P1 and P3 buffers meet the minimum size
                                003F   307          ;   requirements
                                003F   308          ;
                50   0000'8F    B0 003F   309          MOVW    #SS$_ILLCNTRFUNC,R0     ; Assume NFB too small
                     51    03   D0 0044   310          MOVL    #NFB$_ERR_P1,R1         ; Qualify the error
                5B   0048'CF    D0 0047   311          MOVL    NET$GL_PTR_P1,R11       ; Get address of NFB
                0044'CF    05   D1 004C   312          CMPL    #5,NET$GL_SIZ_P1        ; Check for legal NFB size
                           74   1A 0051   313          BGTRU   100$                    ; If GTRU too small
                    0000'CF     D4 0053   314          CLRL    NET$GL_PM_OUT           ; Init output item count
                    0034'CF     D5 0057   315          TSTL    NET$GL_SIZ_P3           ; Was there a P3 buffer?
                               0C   12 005B   316          BNEQ    20$                     ; If EQL no
          0038'CF   0114'CF    9E 005D   317          MOVAB   DUMMY_P3,NET$GL_PTR_P3  ; Use dummy P3
               0034'CF    02   D0 0064   318          MOVL    #2,NET$GL_SIZ_P3        ; ...and setup its size
                     51    05   D0 0069   319 20$:     MOVL    #NFB$_ERR_P3,R1         ; Assume P3 buffer is too small
               0034'CF    02   D1 006C   320          CMPL    #2,NET$GL_SIZ_P3        ; Is P3 buffer big enough ?
                           54   1A 0071   321          BGTRU   100$                    ; If GTRU then no
                    0038'DF     B4 0073   322          CLRW    @NET$GL_PTR_P3          ; Init P3 "buffer"
                                0077   323          ;
                                0077   324          ;   Dispatch to action routine. Mark the IPR for buffer writeback
                                0077   325          ;   if the action routine was successful or if R0 = SS$_RESULTOVF
                                0077   326          ;
                  CD'AF    00   FB 0077   327          CALLS   #0,B^DISPATCH           ; Disptach to process the request
               0000'CF     50   B0 007B   328          MOVW    R0,NET$GQ_USR_STAT      ; Set I/O status
                           0A   12 0080   329          BNEQ    33$                     ; Was the status code zero?
               50   0000'8F     3C 0082   330          MOVZWL  #SS$_ABORT,R0           ; If so there's a bug, use catch-all
               0000'CF     50   B0 0087   331          MOVW    R0,NET$GQ_USR_STAT      ; Set I/O status
                        07 50   E8 008C   332 33$:     BLBS    R0,35$                  ; If LBS successful
               0000'8F     50   B1 008F   333          CMPW    R0,#SS$_RESULTOVF       ; Result overflow ?
                           30   12 0094   334          BNEQ    60$                     ; If not, branch
                     52    8B   9A 0096   335 35$:     MOVZBL  (R11)+,R2               ; Get NFB fct
          52   0148'CF42    9A 0099   336          MOVZBL  WRTBCKFCT[R2],R2        ; Get write-back buffer i.d.'s
                           25   13 009F   337          BEQL    60$                     ; If EQL then none
                04 52    01   E0 00A1   338          BBS     #1,R2,40$               ; If BS P1 buffer is to be written back
                    0138'DF     B4 00A5   339          CLRW    @P1_ABD_CNT             ; Prevent write-back of P1 buffer
                04 52    02   E0 00A9   340 40$:     BBS     #2,R2,45$               ; If BS P2 buffer is to be written back
                    0134'DF     B4 00AD   341          CLRW    @P2_ABD_CNT             ; Clear descriptor count field
                08 52    04   E0 00B1   342 45$:     BBS     #4,R2,50$               ; If BS P4 buffer is to be written back
                    0130'DF     B4 00B5   343          CLRW    @P4_ABD_CNT             ; Clear descriptor count field
```

```
                0038'DF   B4   00B9   344          CLRW    @NET$GL_PTR_P3          ; Clear count of bytes returned via P4
                          00BD   345 50$:  SETBIT  IRP$V_FUNC,IRP$W_STS(R3) ; Mark IRP for writeback
           6B   0000'CF   D0   00C1   346          MOVL    NET$GL_PM_OUT,(RT1)     ; Update NFB parameter
                          05   00C6   347 60$:  RSB                             ; Return
                               00C7   348
                               00C7   349 ;
                               00C7   350 ; Error detected in argument list
                               00C7   351 ;
                               00C7   352
      0000'CF   50   7D   00C7   353 100$:  MOVQ    R0,NET$GQ_USR_STAT      ; Store final IOSB
                          05   00CC   354          RSB
```

```
                                00CD   356 ;
                                00CD   357 ; Dispatch to proper function processor
                                00CD   358 ;
                                00CD   359 DISPATCH:
                          0828  00CD   360         .WORD   ^M<R3,R5,R11>           ; ENTRY
                                00CF   361
                    52    8B    9A   00CF   362         MOVZBL  (R11)+,R2               ; Get NFB function
          0004'CF   6B    D0   00D2   363         MOVL    (R11),NET$GL_PM_IN      ; Save NFB parameter
                26  52    91   00D7   364         CMPB    R2,#NFB$C_FC_MAX        ; Within range ?
                    60    1A   00DA   365         BGTRU   ILLFCT                  ; Illegal NFB fct if GTRU
                                00DC   366
0140'CF   0010'CF42    7D   00DC   367         MOVQ    PRV_Q_REQ[R2],QUAD_BUF  ; Get user's privilege mask
                      1D    E1   00E4   368         BBC     #PRV$V_BYPASS,-         ; Branch if user doesn't have BYPASS
                   06 40 A3          00E6   369                 IRP$Q_NT_PRVMSK(R3),10$
                                00E9   370         SETBIT  NET$V_BYPASS,NET$GL_FLAGS ; Remember privilege
                                00EF   371         ;
                                00EF   372         ;  #64 is illegal in the FFS instruction -- this logic must be updated
                                00EF   373         ;  to include both parts of the mask when privilege bits 32-63 are
                                00EF   374         ;  defined.
                                00EF   375         ;
50  0140'CF    20   00    EA   00EF   376 10$:    FFS     #0,#32,QUAD_BUF,R0      ; Get required privilege
                      0D    13   00F6   377         BEQL    30$                     ; If EQL none left
                                00F8   378         CLRBIT  R0,QUAD_BUF             ; Clear the bit for loop
          EC 40 A3    50    E0   00FE   379         BBS     R0,IRP$Q_NT_PRVMSK(R3),10$ ; If BS user has privilege
                      2E    11   0103   380         BRB     NO_PRV                  ; Else report error
                      5A    7C   0105   381 30$:    CLRQ    R10                     ; Init CNF,CNR pointers
                 32'AF    9F   0107   382         PUSHAB  B^40$                   ; Setup return address
                                010A   383         $DISPATCH   R2,-                ; Dispatch on NFB function
                                010A   384         <-
                                010A   385             <NFB$C_LOGEVENT,  NET$LOG_EVENT>,-
                                010A   386             <NFB$C_READEVENT, NET$READ_EVENT>,-
                                010A   387
                                010A   388             <NFB$C_DECLNAME,  DCL_NAME>,-
                                010A   389             <NFB$C_DECLOBJ,   DCL_OBJECT>,-
                                010A   390             <NFB$C_DECLSERV,  DCL_SERVER>,-
                                010A   391
                                010A   392             <NFB$C_FC_SET,    CTL_DATABASE>,-
                                010A   393             <NFB$C_FC_CLEAR,  CTL_DATABASE>,-
                                010A   394             <NFB$C_FC_SHOW,   CTL_DATABASE>,-
                                010A   395             <NFB$C_FC_DELETE, CTL_DATABASE>,-
                                010A   396             <NFB$C_FC_ZERCOU, CTL_DATABASE>,-
                                010A   397         >
                      0A    11   0130   398         BRB     ILLFCT                  ; IO$_ACPCONTROL function unkown
                      04    0132   399 40$:    RET
                                0133   400
0004'CF   50    D0    0133   401 NO_PRV: MOVL    R0,NET$GQ_USR_STAT+4    ; Qualify error
     50   00'   3C    0138   402         MOVZWL  S^#SS$_NOPRIV,R0        ; Set status
                      04    013B   403         RET                             ; Return to dispatcher
                                013C   404
                      01    D0   013C   405 ILLFCT: MOVL    #NFB$_ERR_FCT,-         ; Qualify error
          0004'CF                 013E   406                 NET$GQ_USR_STAT+4
50  0000'8F    3C    0141   407         MOVZWL  #SS$_ILLCNTRFUNC,R0     ; Illegal ACP control function
                      04    0146   408         RET                             ; Return to dispatcher
```

```
                                0147   410          .SBTTL  Declare Name or Object
                                0147   411
                                0147   412          .ENABL  LSB
                                0147   413
                                0147   414 DCL_OBJECT:                                ; "DECLARE OBJECT" action routine
                                0147   415          ASSUME  NET$C_MAX_OBJ  LE  255
                                0147   416          ASSUME  DUMMY_P2_LNG   GE    8  ; DUMMY_P2 buffer will hold object name
                                0147   417
            003C'CF    D5   0147   418          TSTL    NET$GL_SIZ_P2                ; Was a P2 specified?
                  46   12   014B   419          BNEQ    10$                          ; If NEQ yes - error
       50   0004'CF    9A   014D   420          MOVZBL  NET$GL_PM_IN,R0              ; Pick up number for name conversion
                  3F   13   0152   421          BEQL    10$                          ; Zero is illegal for DECLARED Objects
  000000FF 8F     50   D1   0154   422          CMPL    R0,#NET$C_MAX_OBJ            ; Is number within allowed range?
                  36   1A   015B   423          BGTRU   10$                          ; If GTRU then out of range
       0150'CF    50   90   015D   424          MOVB    R0,CTL_DCLZNA                ; Save object number as ZNA string
       53  004C'CF    9E   0162   425          MOVAB   DUMMY_P2,R3                  ; Get pointer to name buffer
       0040'CF    53   D0   0167   426          MOVL    R3,NET$GL_PTR_P2             ; Setup pointer to it
       003C'CF    53   CE   016C   427          MNEGL   R3,NET$GL_SIZ_P2             ; Bias the name's size
  83    5F4A424F 8F   D0   0171   428          MOVL    #^A"OBJ ",(R3)+              ; Start building object name
                FE85'   30   0178   429          BSBW    NET$BIN2ASC                  ; Append converted object number
       003C'CF    53   C0   017B   430          ADDL    R3,NET$GL_SIZ_P2             ; Calculate name's size
                  57   7C   0180   431          CLRQ    R7                           ; Object name portion is null in ZNA
                               0182   432                                           ; string for numbered objects
                  16   11   0182   433          BRB     DCL_COMMON                   ; Finish in common code
                               0184   434
                               0184   435 DCL_NAME:                                   ; "DECLARE NAME" action routine
       58  0040'CF    D0   0184   436          MOVL    NET$GL_PTR_P2,R8             ; Get string pointer
       57  003C'CF    D0   0189   437          MOVL    NET$GL_SIZ_P2,R7             ; And its size
            0C    57   D1   018E   438          CMPL    R7,#NET$C_MAXOBJNAM          ; Can't be bigger than this
                  03   1B   0191   439          BLEQU   20$                          ; If GTRU the QIO error
                0099   31   0193   440 10$:     BRW     BADPARAM1                    ;!better error code needed?
                               0196   441
            0150'CF    94   0196   442 20$:     CLRB    CTL_DCLZNA                   ; Make obj number be 0
                               019A   443
                               019A   444 DCL_COMMON:                                ; Common code for obj and names
                               019A   445          ;
                               019A   446          ;
                               019A   447          ;  INPUTS:      R7,R8   Descriptor of "name" portion of ZNA field
                               019A   448          ;
                               019A   449          ;       NET$GL_PTR_P2   Descriptor of actual object name
                               019A   450          ;       NET$GL_SIZ_P2
                               019A   451          ;
                               019A   452          ;
0151'CF    68    57   28   019A   453          MOVC3   R7,(R8),CTL_DCLZNA+1         ; Finish building the ZNA string
                  57   D6   01A0   454          INCL    R7                           ; Account for the object number
       58  0150'CF    9E   01A2   455          MOVAB   CTL_DCLZNA,R8                ; Point to it
       0148'CF    57   7D   01A7   456          MOVQ    R7,CTL_Q_DCLZNA              ; Save object's ZNA descriptor
       51  0000'CF    D0   01AC   457          MOVL    NET$GL_SAVE_UCB,R1           ; Get UCB address
       50  0000'8F    3C   01B1   458          MOVZWL  #SS$_NOMBX,R0                ; Assume error
            60  A1    D5   01B6   459          TSTL    UCB$L_AMB(R1)                ; Is there an associated mailbox?
                  77   18   01B9   460          BGEQ    100$                         ; If GEQ then no
       5B  0000'CF    D0   01BB   461          MOVL    NET$GL_CNR_OBI,R11           ; Point the OBI root block
                               01C0   462          $SEARCH egl,obi,s,zna                ; Locate matching object in database
              10 50    E9   01CD   463          BLBC    R0,40$                       ; If LBC no its not there
                               01D0   464          $GETFLD obi,l,ucb                    ; See if name has been declared
              51 50    E8   01DB   465          BLBS    R0,BADPARAM1                 ; If LBS yes - error
                  05   11   01DE   466          BRB     50$                          ; Continue
```

```
                     01E0   467 40$:    ;
                     01E0   468          ;
                     01E0   469          ;   The OBI doesn't exist in the database, create one
                     01E0   470          ;
          51   10      01E0   471          BSBB    CREATE_OBI                  ; Create OBI entry
        4D 50   E9      01E2   472          BLBC    R0,100$                     ; Exit on error
                     01E5   473 50$:     ;
                     01E5   474          ;   Mark OBI as "declared"
                     01E5   475          ;
56   0000'CF   D0      01E5   476          MOVL    NET$GL_SAVE_IRP,R6          ; Get the IRP address
  58   1C A6   D0      01EA   477          MOVL    IRP$L_UCB(R6),R8            ; Get UCB address...
                     01EE   478          $PUTFLD obi,l,ucb                   ; ...and store it in the OBI block
  50   0C A6   D0      01F9   479          MOVL    IRP$L_PID(R6),R0            ; Get the declarer's PID...
 00000000'GF   16      01FD   480          JSB     G^EXE$IPID_TO_EPID          ; ...convert to EPID format...
     58   50   D0      0203   481          MOVL    R0,R8                       ; ...
                     0206   482          $PUTFLD obi,l,pid                   ; ...and store it in the OBI block
  58   28 A6   3C      0211   483          MOVZWL  IRP$W_CHAN(R6),R8           ; Get the declarer's channel...
                     0215   484          $PUTFLD obi,l,chn                   ; ...and store it in the OBI block
                     0220   485          ;
                     0220   486          ;
                     0220   487          ;   Send any pending connects to the declaring process
                     0220   488          ;
57   0148'CF   7D      0220   489          MOVQ    CTL_Q_DCLZNA,R7             ; Get ZNA descriptor
        FDD8'   30      0225   490          BSBW    NET$SCAN_FOR_ZNA            ; Send pending connects to object
50   0000'8F   3C      0228   491          MOVZWL  #SS$_NORMAL,R0              ; Return success if we made it this far
          03   11      022D   492          BRB     100$                        ; Return with R0
                     022F   493
                     022F   494 BADPARAM1:
     50   00'   D0      022F   495          MOVL    S^#SS$_BADPARAM,R0          ; Bad parameter
                05      0232   496 100$:    RSB                                 ; Return
                     0233   497
                     0233   498                      .DSABL  LSB
                     0233   499
                     0233   500
                     0233   501 CREATE_OBI:                                    ; Create OBI and insert it into the list
                     0233   502          ;
                     0233   503          ;   This subroutine in required so that the  "utility buffer"  acquired
                     0233   504          ;   by the NET$GETUTLBUF co-routine will be released in a timely manner.
                     0233   505          ;
        FDCA'   30      0233   506          BSBW    NET$GETUTLBUF               ; Get permission to use utility buffer
        FDC7'   30      0236   507          BSBW    CNF$INIT_UTL                ; Init "utility buffer" as a CNF
58   0040'CF   D0      0239   508          MOVL    NET$GL_PTR_P2,R8            ; Get object name string pointer
57   003C'CF   D0      023E   509          MOVL    NET$GL_SIZ_P2,R7            ; And its size
                     0243   510          $PUTFLD obi,s,nam                   ; Store by object name
58   0150'CF   9A      024E   511          MOVZBL  CTL_DCLZNA,R8               ; Setup the object number...
                     0253   512          $PUTFLD obi,l,num                   ; ...and store it in the CNF
          56   D4      025E   513          CLRL    R6                          ; No "old" CNF
        FD9D'   30      0260   514          BSBW    CNF$INSERT                  ; Try to put block into list
        0E 50   E9      0263   515          BLBC    R0,10$                      ; If LBC then failure
                     0266   516          $CLRFLD obi,v,set                   ; Not created via a "set" QIO
     50   00'   D0      0271   517          MOVL    S^#SS$_NORMAL,R0            ; Indicate success
                05      0274   518 10$:     RSB                                 ; Release utility buffer
                     0275   519
```

```
                     0275   521          .SBTTL  Declare server process available for new connect
                     0275   522 ;+
                     0275   523 ; DCL_SERVER - Process request from a server for another connect
                     0275   524 ;
                     0275   525 ; This QIO can be issued by a nonprivileged process to indicate that
                     0275   526 ; it is willing to process another incoming connect, as long as the
                     0275   527 ; new connect matches the user context currently set in the server.
                     0275   528 ;
                     0275   529 ; Inputs:
                     0275   530 ;
                     0275   531 ;       R3 = IRP address
                     0275   532 ;
                     0275   533 ; Outputs:
                     0275   534 ;
                     0275   535 ;       None
                     0275   536 ;-
                     0275   537 DCL_SERVER:
                     0275   538 ;
                     0275   539 ; Find the database entry associated with this server process.  If not
                     0275   540 ; found in the SPI database, then it wasn't created by us.
                     0275   541 ;
5B   0000'CF   DO    0275   542          MOVL    NET$GL_CNR_SPI,R11      ; Get address of SPI root block
         5A    D4    027A   543          CLRL    R10                     ; Start at beginning of list
   58    0C A3   DO    027C   544          MOVL    IRP$L_PID(R3),R8        ; Get PID of requestor
                     0280   545          $SEARCH eql,spi,l,pid           ; Find it in the database
         03 50   E8    028D   546          BLBS    R0,10$                  ; If not found,
          0084   31    0290   547          BRW     100$                    ; report "illegal request"
                     0293   548 ;
                     0293   549 ; Store the IRP address in the database entry, to be later retrieved when
                     0293   550 ; an incoming connect comes in which this server can handle.  If there is
                     0293   551 ; already an IRP waiting for this process, then return an error.
                     0293   552 ;
                     0293   553 10$:    $GETFLD spi,l,irp               ; Is there already an IRP waiting?
        76 50    E8   029E   554          BLBS    R0,100$                 ; If so, "duplicate request"
     58    53    DO   02A1   555          MOVL    R3,R8                   ; Set IRP address
       FD59'    30   02A4   556          BSBW    CNF$PUT_FIELD           ; Save IRP in database
   0000'CF      D4   02A7   557          CLRL    NET$GL_SAVE_IRP         ; Do not post IRP on return
50 0000'CF      DO   02AB   558          MOVL    NET$GL_PTR_VCB,R0       ; Get RCB address
     0C A0      B6   02B0   559          INCW    RCB$W_TRANS(R0)         ; Account for tucked-away IRP
  58   28 A3    3C   02B3   560          MOVZWL  IRP$W_CHAN(R3),R8       ; Get channel number
                     02B7   561          $PUTFLD spi,l,chn               ; Store it
                     02C2   562 ;
                     02C2   563 ; If this server was supposed to be handling a logical link, then it must
                     02C2   564 ; have failed to confirm the previous logical link for some reason.  In
                     02C2   565 ; this case, notify NETDRIVER to break any previous links intended for the
                     02C2   566 ; previous incarnation of the server.
                     02C2   567 ;
                     02C2   568          $GETFLD spi,s,ncb               ; Was a link being processed already?
        14 50    E9   02CD   569          BLBC    R0,20$                  ; Branch if not
                     02D0   570          $GETFLD spi,l,pid               ; Get the PID
     51    58    DO   02DB   571          MOVL    R8,R1                   ; Set to proper register for call
     52    26    DO   02DE   572          MOVL    #NET$C_DR_EXIT,R2       ; Set "network partner exited"
       FD1C'    30   02E1   573          BSBW    NET$SERVER_FAIL         ; Notify NETDRIVER that server done
                     02E4   574 ;
                     02E4   575 ; Clear out the fields relevant only to the last connect handled by this
                     02E4   576 ; process, since we know it is now done handling it.
                     02E4   577 ;
```

```
                        02E4    578 20$:    $CLRFLD spi,s,sfi                   ; Clear procedure filespec
                        02EF    579         $CLRFLD spi,s,ncb                   ; Clear NCB
                        02FA    580         $CLRFLD spi,s,pnm                   ; Clear process name
                        0305    581 ;
                        0305    582 ; If the initial connect request hasn't been accepted yet, then assume
                        0305    583 ; the process declared itself ready before getting to the point where
                        0305    584 ; the accepting procedure was run.  So, satisfy the DECLSERV request now
                        0305    585 ; so that first connect will be accepted.
                        0305    586 ;
                        0305    587         $GETFLD spi,l,pid                   ; Get the PID again
                FCED'   30      0310    588         BSBW    NET$RESEND_SERVER   ; Send pending connects to server
             50   01    D0      0313    589         MOVL    #1,R0               ; Success
                        05      0316    590         RSB
                                0317    591
     50  00000000'8F    D0      0317    592 100$:   MOVL    #SS$_ILLCNTRFUNC,R0 ; Return error
                        05      031E    593         RSB
```

```
                     031F    595          .SBTTL  Cancel I/O
                     031F    596 ;++
                     031F    597 ;
                     031F    598 ; NET$DRV_CANCEL - Process cancel function from driver
                     031F    599 ; NET$ACP_CANCEL - Process cancel function from exec
                     031F    600 ;
                     031F    601 ; INPUTS:
                     031F    602 ;       NET$GL_SAVE_IRP - IRP address (NET$ACP_CANCEL)
                     031F    603 ;       R11 - pointer to PID and CHN (NET$DRV_CANCEL)
                     031F    604 ;
                     031F    605 ;--
                     031F    606 NET$DRV_CANCEL::
016E'CF    8B   D0   031F    607          MOVL    (R11)+,CANCEL_L_PID     ; Get the PID
017E'CF    6B   B0   0324    608          MOVW    (R11),CANCEL_W_CHN      ; Get the channel
           11   11   0329    609          BRB     CANCEL_COMMON           ; Finish in common code
                     032B    610
                     032B    611 NET$ACP_CANCEL::
   53   0000'CF   D0   032B    612          MOVL    NET$GL_SAVE_IRP,R3          ; Get the IRP
016E'CF    0C A3   D0   0330    613          MOVL    IRP$L_PID(R3),CANCEL_L_PID  ; Get the PID
017E'CF    28 A3   B0   0336    614          MOVW    IRP$W_CHAN(R3),CANCEL_W_CHN ; Get the channel
                     033C    615
                     033C    616 CANCEL_COMMON:
                     033C    617          ;
                     033C    618          ;  Search known object list to see if cancelling process is a known
                     033C    619          ;  object  that should be removed.
                     033C    620          ;
   5B   0000'CF   D0   033C    621          MOVL    NET$GL_CNR_OBI,R11      ; Get known object list root address
           5A   D4   0341    622          CLRL    R10                     ; No CNF yet
   50   016E'CF   D0   0343    623 10$:     MOVL    CANCEL_L_PID,R0         ; Get the match value
    00000000'GF   16   0348    624          JSB     G^EXE$IPID_TO_EPID      ; Convert it to EPID format
        58   50   D0   034E    625          MOVL    R0,R8                   ; Set up register for $SEARCH
                     0351    626          $SEARCH eql,obi,l,pid           ; Set to match on EPID
          46 50   E9   035E    627          BLBC    R0,20$                  ; If LBC no match
                     0361    628          $GETFLD obi,l,chn               ; Get the channel
58   017E'CF   B1   036C    629          CMPW    CANCEL_W_CHN,R8         ; Channels match?
          D0   12   0371    630          BNEQ    10$                     ; If NEQ no - try next
                     0373    631          $CLRFLD obi,l,ucb               ; Clear the UCB field
                     037E    632          $CLRFLD obi,l,pid               ; Clear the PID field
                     0389    633          $CLRFLD obi,l,chn               ; Clear the CHN field
                     0394    634          $GETFLD obi,v,set               ; Was the "set" QIO used to create OBI?
        A1 58   E8   039F    635          BLBS    R8,10$                  ; If LBS yes, leave it in the database
         FC5B'  30   03A2    636          BSBW    CNF$DELETE              ; Else attempt to mark it for delete
           9C   11   03A5    637          BRB     10$                     ; Loop
         FC56'  30   03A7    638 20$:     BSBW    CNF$PURGE               ; Drain queue of all CNFs marked for
                     03AA    639                                          ; delete
                     03AA    640          ;
                     03AA    641          ;  Search server process database, and clean up any DECLSERV requests
                     03AA    642          ;  that happen to be associated with the cancelling channel.
                     03AA    643          ;
5B   0000'CF   D0   03AA    644          MOVL    NET$GL_CNR_SPI,R11      ; Get Server Process root
           5A   D4   03AF    645          CLRL    R10                     ; Start at beginning
51   0162'CF   9E   03B1    646          MOVAB   SPI_CANCEL_SRCH,R1      ; Point to multiple search key list
         FC47'  30   03B6    647          BSBW    CNF$SEARCH              ; Find the block
        27 50   E9   03B9    648          BLBC    R0,40$                  ; If LBC no match
                     03BC    649          $GETFLD spi,l,irp               ; Waiting DECLSERV IRP?
        19 50   E9   03C7    650          BLBC    R0,40$                  ; Branch if no IRP waiting
         FC33'  30   03CA    651          BSBW    CNF$CLR_FIELD           ; Clear it from entry
```

```
                53    58  D0  03CD   652          MOVL    R8,R3                           ; Copy IRP address
          38 A3   0000'8F  3C  03D0   653          MOVZWL  #SS$_ABORT,IRP$L_IOST1(R3) ; Set abort status
              55    1C A3  D0  03D6   654          MOVL    IRP$L_UCB(R3),R5                ; Get UCB address
        00000000'GF        16  03DA   655          JSB     G^COM$POST                      ; Complete the request
                FC1D'      30  03E0   656          BSBW    NET$DEC_TRANS                   ; Account for completed transaction
                           05  03E3   657 40$:     RSB                                     ; Dore
```

```
03E4    659          .SBTTL  CTL_DATABASE - Process database QIOs
03E4    660 ;
03E4    661 ;  Above the QIO interface each database appears to consist of a number of
03E4    662 ;  entries, e.g., node FRED, node 33, object FAL, etc.  Each entry contains a
03E4    663 ;  number of parameters, e.g., a node name, a node address, and object number,
03E4    664 ;  a line cost, etc.
03E4    665 ;
03E4    666 ;  Below the QIO interface each database consists of a number of CNF blocks,
03E4    667 ;  one CNF block per entry.  Each CNF block consists of a number of fields, one
03E4    668 ;  field per parameter.  Although many CNF "fields" are actually data cells
03E4    669 ;  found within the CNF block, some are actually indexes of action routines
03E4    670 ;  which calculate the field's value.  These action routine "fields" are read-
03E4    671 ;  only.  An example of such a field is the number of hops to a given node.
03E4    672 ;
03E4    673 ;  Each field has an "i.d." and a "value".  The field i.d. serves as an index
03E4    674 ;  into the semantic table portion of that database's CoNfiguration Root block
03E4    675 ;  (CNR).  The semantic table contains information for each field describing
03E4    676 ;  the field format (longword, string, etc), where in the CNF it may be found
03E4    677 ;  or which action routine to call to calculate its value, and miscellaneous
03E4    678 ;  information such as whether it is read-write, read-only, etc.
03E4    679 ;
03E4    680 ;  A generic field defined for all databases is the NFB$C_WILDCARD field.
03E4    681 ;  It always matches any entry it is compared against; this field is used to
03E4    682 ;  facilitate database searches where it is desirable to find all CNFs.  It
03E4    683 ;  is equivalent to not specifying any SEARCH key at all.
03E4    684 ;
03E4    685 ;  There are actually two types of CNF blocks: The "actual" CNF blocks are CNFs
03E4    686 ;  which exist in the database even while not being referenced -- these blocks
03E4    687 ;  are created as a consequence of some IO$_ACPCONTROL QIO.  The "phantom" CNF
03E4    688 ;  blocks are CNFs which exist only while being referenced -- these blocks
03E4    689 ;  represent things known to the ACP but for which no database entry was ever
03E4    690 ;  defined.  As an example, a "phantom" CNF is created while the ACP is
03E4    691 ;  obtaining information about a node which was made known to the ACP via a
03E4    692 ;  routing message but for which was never explicitly defined by the Network
03E4    693 ;  Management layer.
03E4    694 ;
```

```
03E4   696 ;
03E4   697 ;  QIOs To Access the NETACP DataBase
03E4   698 ;
03E4   699 ;  The following control QIOs provide access to the NETACP data base.  The
03E4   700 ;  factors which influenced the design of these QIOs were:
03E4   701 ;
03E4   702 ;
03E4   703 ;    o  To provide a common mechanism to access all parts of the database
03E4   704 ;       in order to simplify programming.
03E4   705 ;
03E4   706 ;    o  To allow the user to utilize a table driven approach.
03E4   707 ;
03E4   708 ;    o  To reduce the proliferation of a series of  ad hoc  QIOs which are
03E4   709 ;       difficult to re-implement if and when the NETACP is modified.
03E4   710 ;
03E4   711 ;
03E4   712 ;  The QIO parameters specific to these functions are:
03E4   713 ;
03E4   714 ;  FUNC = #IO$_ACPCONTROL.
03E4   715 ;  IOSB = Address of the optional IOSB.
03E4   716 ;
03E4   717 ;         Parameters P1 thru P5 each pass the address of a quadword
03E4   718 ;         buffer descriptor.  The buffers are used as follows:
03E4   719 ;
03E4   720 ;  P1   = Supplies the Network Qio Control block (NFB).
03E4   721 ;  P2   = Supplies the search key block.
03E4   722 ;  P3   = Number of bytes returned in the P4 buffer.
03E4   723 ;  P4   = Returns or supplies the specified parameter values.
03E4   724 ;
03E4   725 ;
03E4   726 ;  Errors returned in the IOSB:
03E4   727 ;
03E4   728 ;  SS$_NOPRIV       User lacks the required privilege.  The second longword of
03E4   729 ;                  the IOSB contains the bit number of the first required
03E4   730 ;                  privilege which the user did not have.
03E4   731 ;
03E4   732 ;  SS$_ILLCNTRFUNC Illegal ACP control function.  The second longword of the
03E4   733 ;                  IOSB contains the reason as follows:
03E4   734 ;
03E4   735 ;  SS$_RESULTOVF   The P4 buffer is too small.
03E4   736 ;
03E4   737 ;  SS$_BADPARAM    One of the field identifiers was unrecognized.  The value of
03E4   738 ;                  the identifier is returned in the second IOSB longword.
03E4   739 ;
03E4   740 ;  SS$_ENDOFFILE   No entries were found which matched a search key.  The field
03E4   741 ;                  i.d. of this search key is returned in the 2nd IOSB longword.
03E4   742 ;
```

```
                        03E4    744
                        03E4    745          .ENABL  LSB
                        03E4    746
                        03E4    747 CTL_DATABASE:                           ; Common Control Qio Processing
56   0048'CF    D0      03E4    748          MOVL    NET$GL_PTR_P1,R6       ; Get base address of NFB
                        03E9    749
                        03E9    750          ;&&
                        03E9    751          ;&&     Kludge to make both the old COLLATE NFBs and the new double
                        03E9    752          ;&&     search key NFBs work with this ACP
                        03E9    753          ;&&
          00000020      03E9    754          NFB$C_CTX_SIZE = 32            ; Accept the lesser of the two sizes
                        03E9    755
  02    08 A6    D1      03E9    756          CMPL    NFB$L_SRCH2_KEY(R6),#2 ; Was old START ID field = COLLATE?
        07       12      03ED    757          BNEQ    2$
                        03EF    758          CLRBIT  NFB$V_NOCTX,NFB$B_FLAGS(R6) ; Force context to be stored
        08 A6    D4      03F3    759          CLRL    NFB$L_SRCH2_KEY(R6)    ; Mark second search key not present
04   003C'CF    D1      03F6    760 2$:      CMPL    NET$GL_SIZ_P2,#4       ; Is a context area present?
        04       14      03FB    761          BGTR    3$                     ; If not, then don't store/fetch context
                        03FD    762          SETBIT  NFB$V_NOCTX,NFB$B_FLAGS(R6) ; No context area to be used
                        0401    763 3$:      ;&&
                        0401    764          ;&&     End of kludge
                        0401    765          ;&&
                        0401    766
                        0401    767
                        0401    768          ;&&
                        0401    769          ;&&     Kludge to make old format control QIO's work with this ACP
                        0401    770          ;&&
                        0401    771 ;        NFB$C_CTX_SIZE = 32            ; Use old value
                        0401    772
                        0401    773 ;        CMPL    NFB$L_MBZ1(R6),#2      ; Was old START ID field = COLLATE?
                        0401    774 ;        BNEQ    4$                     ; Branch if not
                        0401    775 ;        CLRBIT  NFB$V_NOCTX,NFB$B_FLAGS(R6) ; Mark context to be stored
                        0401    776 ;        BRB     5$
                        0401    777 ;4$:     SETBIT  NFB$V_NOCTX,NFB$B_FLAGS(R6) ; Mark do not store context
                        0401    778 ;5$:     CLRL    NFB$L_MBZ1(R6)         ; Clear obsolete START ID field
                        0401    779          ;&&
                        0401    780          ;&&     End of kludge
                        0401    781          ;&&
                        0401    782
011C'CF  0030'CF    D0  0401    783          MOVL    NET$GL_PTR_P4,PTR_L_P4 ; Make copy of P4 descriptor
0118'CF  002C'CF    D0  0408    784          MOVL    NET$GL_SIZ_P4,SIZ_L_P4
                        040F    785          ;
                        040F    786          ;   Verify that the NFB (P1) buffer is large enough and that all fields
                        040F    787          ;   have proper values.  This excludes the field i.d. list at the end
                        040F    788          ;   which is checked separately
                        040F    789          ;
      51    03    D0      040F    790          MOVL    #NFB$_ERR_P1,R1        ; Preset error qualifier
    52  0044'CF   D0      0412    791          MOVL    NET$GL_SIZ_P1,R2       ; Get size of P1 buffer
      52    10    C2      0417    792          SUBL    #NFB$L_FLDID,R2        ; Subtract all but the field i.d. list
                        041A    793                                         ; size
            71    15      041A    794          BLEQ    ILL_FUNC               ; If LEQ then too small, report error
                        041C    795 ;        TSTL    NFB$L_MBZ1(R6)         ; MBZ field non-zero?
                        041C    796 ;        BNEQ    ILL_FUNC               ; Report error if so
                        041C    797 ;        TSTW    NFB$W_MBZ2(R6)         ; MBZ field non-zero?
                        041C    798 ;        BNEQ    ILL_FUNC               ; Report error if so
        0D A6    95      041C    799          TSTB    NFB$B_MBZ1(R6)         ; MBZ field non-zero?
            6C   12      041F    800          BNEQ    ILL_FUNC               ; Report error if so
```

```
                51    09    D0  0421    801          MOVL    #NFB$_ERR_CELL,R1       ; Assume illegal cell size
              01    0E A6    B1  0424    802          CMPW    NFB$W_CELL_SIZE(R6),#1  ; Cell size must either be GEQU 2, or
                                0428    803                                          ; EQL 0 (indicating no fixed cell size)
                      63    13  0428    804          BEQL    ILL_FUNC                ; If EQL then illegal cell size
                51    0A    D0  042A    805          MOVL    #NFB$_ERR_OPER,R1       ; Assume illegal OPER value specified
                  03 A6    91  042D    806          CMPB    NFB$B_OPER(R6),-        ; Is it out of range?
                      03            0430    807                  #NFB$C_OP_MAXFCT    ;
                      5A    1A  0431    808          BGTRU   ILL_FUNC                ; If GTRU then yes, report error
                                0433    809          ;
                                0433    810          ;
                                0433    811          ;   Find the CNR (semantic table) according for the database type.
                                0433    812          ;
                                0433    813          ;
                51    02    D0  0433    814          MOVL    #NFB$_ERR_DB,R1         ; Preset error qualifier
              5B    02 A6    9A  0436    815          MOVZBL  NFB$B_DATABASE(R6),R11  ; Get the database i.d.
                      51    13  043A    816          BEQL    ILL_FUNC                ; If EQL then no such database
                   1B    5B    91  043C    817          CMPB    R11,#NFB$C_DB_MAX       ; Within range?
                      4C    1A  043F    818          BGTRU   ILL_FUNC                ; If GTRU then out of range
          03 0000'CF    5B    E1  0441    819          BBC     R11,X25_DB_MASK,10$     ; If BC then not exclusively an X.25
                                0447    820                                          ; database
                  04DA    31  0447    821          BRW     REISSUE_X25             ; Re-issue QIO to X25 ACP
           5B   0000'CF4B    D0  044A    822 10$:     MOVL    NET$AL_CNR_TAB[R11],R11 ; Get pointer to the root block (CNR)
                                0450    823          ;
                                0450    824          ;
                                0450    825          ;   Setup pointer to the count of CNF's successfully processed.  This
                                0450    826          ;   counter is found in the first longword of the P2 buffer.  Update
                                0450    827          ;   the internal P2 buffer descriptor.
                                0450    828          ;
                                0450    829          ;
                51    04    D0  0450    830          MOVL    #NFB$_ERR_P2,R1         ; Assume P2 is too small
        0124'CF    0040'CF    D0  0453    831          MOVL    NET$GL_PTR_P2,PTR_CNFCNT ; Save pointer to counter cell
             003C'CF    04    C2  045A    832          SUBL    #4,NET$GL_SIZ_P2        ; Account for bytes used
                      2C    19  045F    833          BLSS    ILL_FUNC                ; If LSS then too small
             0040'CF    04    C0  0461    834          ADDL    #4,NET$GL_PTR_P2        ; Advance P4 pointer
                 0124'DF    D4  0466    835          CLRL    @PTR_CNFCNT             ; Zero the P4 count field
                                046A    836          ;
                                046A    837          ;   Verify that all field IDs in the NFB are known.
                                046A    838          ;
                51    03    D0  046A    839          MOVL    #NFB$_ERR_P1,R1         ; Assume NFB is too small
                04    52    D1  046D    840          CMPL    R2,#4                   ; At least one field ID specified?
                      1B    19  0470    841          BLSS    ILL_FUNC                ; If not, return an error
                03    52    D3  0472    842          BITL    R2,#^B11                ; Does NFB end on longword boundary?
                      16    12  0475    843          BNEQ    ILL_FUNC                ; If not, return an error
             55    10 A6    9E  0477    844          MOVAB   NFB$L_FLDID(R6),R5      ; Get address of first field i.d.
                59    85    D0  047B    845 20$:     MOVL    (R5)+,R9                ; Get next field
                                047E    846          ASSUME  NFB$C_ENDOFLIST EQ 0    ; Field terminator value
                      20    13  047E    847          BEQL    30$                     ; If EQL then at end of list
                   FB7D'    30  0480    848          BSBW    CNF$VERIFY              ; Make sure the field i.d. is valid
                  12 50    E9  0483    849          BLBC    R0,BAD_PARAM            ; Branch if invalid field detected
                52    04    C2  0486    850          SUBL    #4,R2                   ; Account for next field
                      15    13  0489    851          BEQL    30$                     ; Branch if end of NFB
                      EE    11  048B    852          BRB     20$                     ; Loop until all fields checked
                                048D    853
                                048D    854          ;
                                048D    855          ;    Some common error return paths
                                048D    856          ;
                                048D    857 ILL_FUNC:                               ; Report "illegal control function"
```

```
                50   0000'8F    3C  048D   858          MOVZWL  #SS$_ILLCNTRFUNC,R0     ; Setup status code
                59     51       D0  0492   859          MOVL    R1,R9                   ; Copy error qualifier
                     00D6       31  0495   860          BRW     200$                    ; Exit
                                    0498   861 BAD_PARAM:                              ; Report "bad parameter"
                50   0000'8F    3C  0498   862          MOVZWL  #SS$_BADPARAM,R0        ; Setup status code
                     00CE       31  049D   863 209$:    BRW     200$                    ; Exit
                                    04A0   864
                                    04A0   865          ;
                                    04A0   866          ;   Setup primary search key descriptor
                                    04A0   867          ;
                51     0B       D0  04A0   868 30$:     MOVL    #NFB$_ERR_SRCH,R1       ; Assume illegal SEARCH KEY i.d.
              59   04 A6       D0  04A3   869          MOVL    NFB$L_SRCH_KEY(R6),R9   ; Get search key i.d.
                       03       12  04A7   870          BNEQ    40$                     ; Branch if specified
                59     01       D0  04A9   871          MOVL    #NFB$C_WILDCARD,R9      ; Use WILDCARD as default search ID
                     00E8       30  04AC   872 40$:     BSBW    GET_P2_KEY              ; Get key value
                   DB 50        E9  04AF   873          BLBC    R0,ILL_FUNC             ; If LBC error
           0008'CF    59        D0  04B2   874          MOVL    R9,NET$GL_SRCH_ID       ; Save i.d. -- it may have been modified
      000C'CF   03 A6        9A  04B7   875          MOVZBL  NFB$B_OPER(R6),NET$GL_OPER ; Save primary comparison type
           0010'CF    57        7D  04BD   876          MOVQ    R7,NET$GQ_SRCH_KEY      ; Copy the key value
                                    04C2   877          ;
                                    04C2   878          ;   Get secondary search key descriptor
                                    04C2   879          ;
                51     0C       D0  04C2   880          MOVL    #NFB$_ERR_SRCH2,R1      ; Assume illegal ID
              59   08 A6       D0  04C5   881          MOVL    NFB$L_SRCH2_KEY(R6),R9  ; Get search key i.d.
                       03       12  04C9   882          BNEQ    42$                     ; Branch if specified
                59     01       D0  04CB   883          MOVL    #NFB$C_WILDCARD,R9      ; Use WILDCARD as default search ID
                     00C6       30  04CE   884 42$:     BSBW    GET_P2_KEY              ; Get key value
                   B9 50        E9  04D1   885          BLBC    R0,ILL_FUNC             ; If LBC error
           0018'CF    59        D0  04D4   886          MOVL    R9,NET$GL_SRCH2_ID      ; Save i.d. -- it may have been modified
      001C'CF   0C A6        9A  04D9   887          MOVZBL  NFB$B_OPER2(R6),NET$GL_OPER2 ; Save secondary comparison type
           0020'CF    57        7D  04DF   888          MOVQ    R7,NET$GQ_SRCH2_KEY     ; Copy the key value
                                    04E4   889          ;
                                    04E4   890          ;   Call any pre-processing routines specifically assigned to the
                                    04E4   891          ;   database specified in the NFB.  These routines handle pre-search
                                    04E4   892          ;   conditions such as normalizing the search key value.
                                    04E4   893          ;
                     FB19'      30  04E4   894          BSBW    CNF$PRE_QIO             ; Preprocess database and SEARCH keys
                                    04E7   895                                          ; before processing the QIO request
                   B3 50        E9  04E7   896          BLBC    R0,209$                 ; If LBC then error
                                    04EA   897          ;
                                    04EA   898          ;   Unless the NFB$V_NOCTX bit is set, the P2 buffer will be
                                    04EA   899          ;   automatically updated with "current position".  The only error
                                    04EA   900          ;   which could prevent this would be the lack of context space in the
                                    04EA   901          ;   P2 buffer.  By checking now that this is at least NFB$C_CTX_SIZE
                                    04EA   902          ;   bytes, then no errors can occur later.
                                    04EA   903          ;
      0A 01 A6     02        E0  04EA   904          BBS     #NFB$V_NOCTX,NFB$B_FLAGS(R6),45$ ; Skip if no update requested
                51     04        D0  04EF   905          MOVL    #NFB$_ERR_P2,R1         ; Assume P2 is too small
                     003C'CF     D1  04F2   906          CMPL    NET$GL_SIZ_P2,-         ; Enough room in the P2 buffer for
                           20            04F6   907                  #NFB$C_CTX_SIZE     ; automatic context area update?
                           94    1F      04F7   908          BLSSU   ILL_FUNC            ; Error if not
                                    04F9   909          ;
                                    04F9   910          ;   Find the entry in the list at which to begin the search.  If the
                                    04F9   911          ;   context value in the P2 buffer is null (string count=0), then
                                    04F9   912          ;   set the CNF pointer to the head of the list.
                                    04F9   913          ;
                5A     5B        D0  04F9   914 45$:     MOVL    R11,R10                 ; Start standard CNF pointer at the
```

```
                        04FC   915                                     ; begining of the database list
                        04FC   916
                        04FC   917 ;&& Kludge to make old START ID NFBs work with this ACP
                        04FC   918 ;&& since old format NFB didn't require a context area on non-collate QIOs
                        04FC   919 ;&& This kludge prevents newer QIOs which want to start at a given position
                        04FC   920 ;&& in the list, but stay there, from working.  Luckily, nobody does this
                        04FC   921 ;&& right now.
4B 01 A6    02    E0    04FC   922          BBS     #NFB$V_NOCTX,NFB$B_FLAGS(R6),50$ ; Skip if no context present
                        0501   923 ;&& End of kludge
                        0501   924
   59    14 AB    D0    0501   925          MOVL    CNR$L_FLD_COLL(R11),R9  ; Get collating field ID of database
         008F     30    0505   926          BSBW    GET_P2_KEY              ; Get descriptor of context
      82 50       E9    0508   927          BLBC    R0,ILL_FUNC             ; If LBC error
         52       D5    050B   928          TSTL    R2                      ; Is key value "null"
         3D       13    050D   929          BEQL    50$                     ; If EQL yes, start at head of list.
003C'CF  52       C0    050F   930          ADDL    R2,NET$GL_SIZ_P2        ; Put descriptor back, so that it
0040'CF  52       C2    0514   931          SUBL    R2,NET$GL_PTR_P2        ; still points to the context area
   02 02 A6       91    0519   932          CMPB    NFB$B_DATABASE(R6),#NFB$C_DB_NDI ; Searching node database?
         1F       12    051D   933          BNEQ    48$                     ; Branch if not
         68       95    051F   934          TSTB    (R8)                    ; Is first (format) byte 0?
         1B       12    0521   935          BNEQ    48$                     ; If not, use seq. search
      01 A8       B5    0523   936          TSTW    1(R8)                   ; Node number non-zero?
         16       13    0526   937          BEQL    48$                     ; If zero, skip optimization
         58       DD    0528   938          PUSHL   R8                      ; Save registers
   7E 01 A8       90    052A   939          MOVB    1(R8),-(SP)             ; Get 2 bytes of node number
   7E 02 A8       90    052E   940          MOVB    2(R8),-(SP)
      58    8E    3C    0532   941          MOVZWL  (SP)+,R8                ; Get last node number processed
         FAC8'    30    0535   942          BSBW    NET$LOCATE_NDI          ; Find previous NDI position
            58 8ED0    0538   943          POPL    R8                      ; Restore registers
      0E 50    E8    053B   944          BLBS    R0,50$                  ; If found, then skip seq. search
50    0000'8F  3C    053E   945 48$:      MOVZWL  #SS$_ENDOFFILE,R0       ; Assume starting CNF can't be found
   51    06    D0    0543   946          MOVL    #NFB$C_OP_FNDPOS,R1     ; Find last CNF whose key value is GEQU
         FAB7'    30    0546   947          BSBW    CNF$KEY_SRCH_EX         ; the key passed in R7/R8
      22 50    E9    0549   948          BLBC    R0,200$                 ; If LBC then not found
                        054C   949          ;
                        054C   950          ;   Process the selected database entries (CNFs).  If the MULT flag
                        054C   951          ;   is set, then continue to search for CNFs until an error is
                        054C   952          ;   detected (most likely ENDOFFILE or P4-buffer-full).
                        054C   953          ;
         00A7    30    054C   954 50$:      BSBW    PROCESS_CNF             ; Process next CNF
      05 50      E9    054F   955          BLBC    R0,60$                  ; If LBC then error
         01      E0    0552   956          BBS     #NFB$V_MULT,-           ; If BS then process next CNF
   F5 01 A6             0554   957                  NFB$B_FLAGS(R6),50$
                        0557   958          ;
                        0557   959          ;   In the case that we are returning more than one entry in the
                        0557   960          ;   P4 buffer (MULT flag is set), then do not return ENDOFFILE
                        0557   961          ;   or RESULTOVF if we have returned at least one entry.
                        0557   962          ;   The user will get ENDOFFILE on the next QIO if he has hit
                        0557   963          ;   the end of the database.  RESULTOVF is a normal condition
                        0557   964          ;   if we are returning as many entries as possible in P4.
                        0557   965          ;
   0124'DF  D5    0557   966 60$:      TSTL    @PTR_CNFCNT             ; Any CNFs successfully processed?
         11       13    055B   967          BEQL    200$                    ; If EQL then no mapping needed
0000'8F  50       B1    055D   968          CMPW    R0,#SS$_ENDOFFILE       ; Did the search fail ?
         07       13    0562   969          BEQL    70$                     ; If so, return normal this time
0000'8F  50       B1    0564   970          CMPW    R0,#SS$_RESULTOVF       ; P4 buffer overflow?
         03       12    0569   971          BNEQ    200$                    ; If neither status, skip it
```

```
                     50   00'  D0  056B   972 70$:    MOVL    S^#SS$_NORMAL,R0        ; Else report success since at least
                                 056E   973                                          ; one entry was processed.
                                 056E   974          ;
                                 056E   975          ;   Update the IOSB image
                                 056E   976          ;
                0000'CF    50    B0  056E   977 200$:   MOVW    R0,NET$GQ_USR_STAT      ; Set status code in IOSB
                      05   50    E8  0573   978         BLBS    R0,205$                 ; If success, don't store qualifier
                0004'CF    59    D0  0576   979         MOVL    R9,NET$GQ_USR_STAT+4    ; Error qualifier if LBC in R0
                     0030'CF    C3  057B   980 205$:   SUBL3   NET$GL_PTR_P4,-         ; Get number of bytes moved to P4
           52        011C'CF        057F   981                   PTR_L_P4,R2           ; buffer
                0038'DF    52    B0  0583   982         MOVW    R2,@NET$GL_PTR_P3       ; Update count in P3 buffer
                0002'CF    52    B0  0588   983         MOVW    R2,NET$GQ_USR_STAT+2    ; Update count in IOSB image
                           0E    E1  058D   984         BBC     #NET$V_PURGE,-          ; If BC then no need to purge database
                  03 0000'CF        058F   985                   NET$GL_FLAGS,210$     ;
                        FA6A'  30   0593   986         BSBW    CNF$PURGE               ; Drain the queue of all CNFs marked
                               0596   987                                              ; for delete.
                               05  0596   988 210$:    RSB                             ; Done
                                   0597   989
                                   0597   990                 .DSABL  LSB
```

```
                                0597   992          .SBTTL  GET_P2_KEY - Get next P2 value
                                0597   993 ;+
                                0597   994 ; GET_P2_KEY - Get next value from P2 buffer
                                0597   995 ;
                                0597   996 ;  INPUTS:      R9      Field i.d. of the key
                                0597   997 ;               R8,R7   Scratch
                                0597   998 ;               R2      Scratch
                                0597   999 ;               R0      Scratch
                                0597  1000 ;
                                0597  1001 ;  OUTPUTS:     R8,R7   Key value/descriptor
                                0597  1002 ;               R9      Field ID
                                0597  1003 ;               R2      Number of bytes in field.  If the field value is "null"
                                0597  1004 ;                       (negative longword value or string with a zero count
                                0597  1005 ;                       field) then R2 is returned as a zero.
                                0597  1006 ;               R0      Status
                                0597  1007 ;               R1      Error qualifier, if an error was returned.
                                0597  1008 ;
                                0597  1009 ;       NET$GL_PTR_P2,SIZ_P2 will be updated to point past value
                                0597  1010 ;               if routine returns successfully.
                                0597  1011 ;---
                                0597  1012 GET_P2_KEY:                                   ; Locate next key in the P2 buffer
        50    00'   D0          0597  1013          MOVL    S^#SS$_NORMAL,R0            ; Assume success
        01    59    D1          059A  1014          CMPL    R9,#NFB$C_WILDCARD          ; "wild card" key ?
              36    13          059D  1015          BEQL    35$                         ; If so, then there is no key value
           FA5E'    30          059F  1016          BSBW    CNF$VERIFY                  ; Is field i.d. valid ?
        51    59    D0          05A2  1017          MOVL    R9,R1                       ; Return field ID in case of error
          4D  50    E9          05A5  1018          BLBC    R0,90$                      ; If LBC then no
              10    ED          05A8  1019          CMPZV   #NFB$V_TYP,-
        02    59    02          05AA  1020                  #NFB$S_TYP,R9,-
                                05AD  1021                  #NFB$C_TYP_STR              ; Is field a string ?
              13    13          05AD  1022          BEQL    10$                         ; If EQL yes
                                05AF  1023          ;
                                05AF  1024          ;   The field is type "bit" or "longword".  In either case the key
                                05AF  1025          ;   value is stored as a longword in the P2 buffer
                                05AF  1026          ;
        52    04    D0          05AF  1027          MOVL    #4,R2                       ; Setup field size
52      003C'CF     D1          05B2  1028          CMPL    NET$GL_SIZ_P2,R2            ; Can it fit?
              37    1F          05B7  1029          BLSSU   60$                         ; Branch if not
58      0040'DF     D0          05B9  1030          MOVL    @NET$GL_PTR_P2,R8           ; Get field value
              13    19          05BE  1031          BLSS    30$                         ; If LSS then field value is "null"
              22    11          05C0  1032          BRB     70$                         ; Continue in common
                                05C2  1033 10$:     ;
                                05C2  1034          ;   The field is type "string".  It is stored in the P2 buffer as a
                                05C2  1035          ;   word of count followed by the string.
                                05C2  1036          ;
02      003C'CF     D1          05C2  1037          CMPL    NET$GL_SIZ_P2,#2            ; P2 buffer big enough for count field
              27    1F          05C7  1038          BLSSU   60$                         ; Branch if not
58      0040'CF     D0          05C9  1039          MOVL    NET$GL_PTR_P2,R8            ; Get pointer to the count field
        57    88    32          05CE  1040          CVTWL   (R8)+,R7                    ; Get count field value
              06    1A          05D1  1041          BGTRU   40$                         ; If GTRU then not "null"
              57    7C          05D3  1042 30$:     CLRQ    R7                          ; Zero value/descriptor
              52    D4          05D5  1043 35$:     CLRL    R2                          ; Indicate "null" field value
              1C    11          05D7  1044          BRB     90$                         ; Take common exit
52      57    02    C1          05D9  1045 40$:     ADDL3   #2,R7,R2                    ; Get total field size
52      003C'CF     D1          05DD  1046          CMPL    NET$GL_SIZ_P2,R2            ; Is the P2 buffer big enough ?
              0C    1F          05E2  1047          BLSSU   60$                         ; Branch if not
003C'CF       52    C2          05E4  1048 70$:     SUBL    R2,NET$GL_SIZ_P2            ; Account for bytes used in P2 buffer
```

```
                0040'CF    52   C0   05E9  1049          ADDL    R2,NET$GL_PTR_P2        ; Advance past bytes used
                           05   11   05EE  1050          BRB     90$
                                     05F0  1051
                      51   04   D0   05F0  1052 60$:     MOVL    #NFB$_ERR_P2,R1         ; Indicate P2 is too small
                           50   D4   05F3  1053          CLRL    R0                      ; Indicate error
                                05   05F5  1054 90$:     RSB                             ; Return status in R0
```

```
                                05F6  1056          .SBTTL  PROCESS_CNF - Process each CNF block
                                05F6  1057 ;+
                                05F6  1058 ;  Process each (or the first) CNF block found which matches the search key
                                05F6  1059 ;
                                05F6  1060 ;   Creating a new CNF
                                05F6  1061 ;   -----------------
                                05F6  1062 ;
                                05F6  1063 ;     o  The SET Qio is used to both create new and modify existing entries.
                                05F6  1064 ;     o  The Qio issuer is not always aware if the entry already exists
                                05F6  1065 ;     o  If the CNF addressed in a SET Qio is not found then a new CNF will be
                                05F6  1066 ;        created only if the SEARCH_KEY is not 'NFB$C_WILDCARD'.  The SEARCH_KEY
                                05F6  1067 ;        value is inserted into the CNF immediately after it is created.  If
                                05F6  1068 ;        this field is not write-able then the returned Qio status code should
                                05F6  1069 ;        convey the meaning "no such entry" (i.e., SS$_ENDOFFILE).
                                05F6  1070 ;
                                05F6  1071 ;        Note that the created CNF may not meet the requirements which allow it
                                05F6  1072 ;        to be inserted into the database.
                                05F6  1073 ;
                                05F6  1074 ;     o  The decision whether or not create a new CNF entry is independent of
                                05F6  1075 ;        the current position in the database traversal.
                                05F6  1076 ;
                                05F6  1077 ;
                                05F6  1078 ; Inputs:
                                05F6  1079 ;
                                05F6  1080 ;        R11 = CNR address
                                05F6  1081 ;        R10 = Address of starting CNF in list.
                                05F6  1082 ;        R6 = NFB address
                                05F6  1083 ;
                                05F6  1084 ;        NET$AL_SRCH_LIST is setup.
                                05F6  1085 ;
                                05F6  1086 ; Outputs:
                                05F6  1087 ;
                                05F6  1088 ;        R0 = Status
                                05F6  1089 ;
                                05F6  1090 ;        R1-R5,R7-R10 are destroyed.
                                05F6  1091 ;-
                                05F6  1092 PROCESS_CNF:                                 ; Process the next database entry
                0128'CF    D4   05F6  1093          CLRL    PTR_OLD_CNF                ; Initialize old CNF address
                     66    91   05FA  1094          CMPB    NFB$B_FCT(R6),-            ; Is this a "SET" Qio?
                                05FC  1095                  #NFB$C_FC_SET              ;
                     62    12   05FD  1096          BNEQ    60$                        ; Branch if not
                                05FF  1097          ;
                                05FF  1098          ;   Find the next CNF for a "set" function
                                05FF  1099          ;
02010012 8F     0008'CF    D1   05FF  1100          CMPL    NET$GL_SRCH_ID,#NFB$C_NDI_ADD ; Searching by node address?
                     0B    13   0608  1101          BEQL    10$                        ; Branch if so
02010010 8F     0008'CF    D1   060A  1102          CMPL    NET$GL_SRCH_ID,#NFB$C_NDI_TAD ; Search by transformed address?
                     19    12   0613  1103          BNEQ    20$                        ; Branch if not - skip it
          00     000C'CF    D1   0615  1104 10$:     CMPL    NET$GL_OPER,#NFB$C_OP_EQL  ; Using equality match?
                     12    12   061A  1105          BNEQ    20$                        ; Branch if not
          58     0014'CF    D0   061C  1106          MOVL    NET$GQ_SRCH_KEY+4,R8       ; Get desired node address
                     0B    13   0621  1107          BEQL    20$                        ; If zero, then skip
                     5A    DD   0623  1108          PUSHL   R10                        ; Save registers
                   F9D8'    30   0625  1109          BSBW    NET$LOCATE_NDI             ; Find previous NDI position
                     5A 8ED0    0628  1110          POPL    R10                        ; Restore registers
                  10 50    E9   062B  1111          BLBC    R0,30$                     ; If not found, then make new one
                                062E  1112                                             ; Else, use seq. search so that loop
```

```
                          062E  1113                                  ; nodes, etc. processed in sequence
50  0000'8F   3C    062E  1114 20$:    MOVZWL  #SS$_ENDOFFILE,R0      ; Preset error code
51  0008'CF   9E    0633  1115         MOVAB   NET$AL_SRCH_LIST,R1    ; Point to search key list
      F9C5'   30    0638  1116         BSBW    CNF$SEARCH_EX          ; Find the next CNF
        6B 50   E8    063B  1117         BLBS    R0,75$                 ; If found, then don't make new one
                      063E  1118         ;
                      063E  1119         ;   Initialize a new CNF entry
                      063E  1120         ;
59  0008'CF   D0    063E  1121 30$:    MOVL    NET$GL_SRCH_ID,R9      ; Get primary search key ID
57  0010'CF   7D    0643  1122         MOVQ    NET$GQ_SRCH_KEY,R7     ; Get primary search key value
      01  59   D1    0648  1123         CMPL    R9,#NFB$C_WILDCARD     ; Did user have particular CNF in mind?
          11    13    064B  1124         BEQL    40$                    ; If EQL no, don't attempt creation
      F9B0'   30    064D  1125         BSBW    NET$GETUTLBUF          ; Claim the utility buffer
      F9AD'   30    0650  1126         BSBW    CNF$INIT_UTL           ; Init the "utility buffer" as a CNF
50  0000'8F   3C    0653  1127         MOVZWL  #SS$_WRITLCK,R0        ; Assume PUT_FIELD error
      F9A5'   30    0658  1128         BSBW    CNF$PUT_FIELD          ; Attempt to store SEARCH KEY
        50 50   E8    065B  1129         BLBS    R0,76$                 ; If LBC then return error to user.
        00D2    31    065E  1130 40$:    BRW     200$                   ; Take common exit
                      0661  1131         ;
                      0661  1132         ;   Find the next CNF for a non-set function
                      0661  1133         ;
50  0000'8F   3C    0661  1134 60$:    MOVZWL  #SS$_ENDOFFILE,R0      ; Preset error code
51  0008'CF   9E    0666  1135         MOVAB   NET$AL_SRCH_LIST,R1    ; Point to search key list
      F992'   30    066B  1136         BSBW    CNF$SEARCH_EX          ; Find the next CNF
        38 50   E8    066E  1137         BLBS    R0,75$                 ; Branch if found
                      0671  1138         ;
                      0671  1139         ;   On a "show" function, if this is a request for a specific
                      0671  1140         ;   node by address, and the node hasn't been "set" in the
                      0671  1141         ;   database, then use the dummy NDI and allow the operation
                      0671  1142         ;   to continue.
                      0671  1143         ;
          66    91    0671  1144         CMPB    NFB$B_FCT(R6),-        ; Is this a SHOW request?
                22    0673  1145                 #NFB$C_FC_SHOW
                E8    12    0674  1146         BNEQ    40$                    ; Branch if not
                5A    D5    0676  1147         TSTL    R10                    ; Did we start from beginning?
                05    13    0678  1148         BEQL    70$                    ; Br if yes, use DUM_NDI if necessary
          5B    5A    D1    067A  1149         CMPL    R10,R11                ; Did we start from root?
                DF    12    067D  1150         BNEQ    40$                    ; Br if no, return error
02010012 8F   0008'CF   D1    067F  1151 70$:    CMPL    NET$GL_SRCH_ID,#NFB$C_NDI_ADD ; Searching by node address?
                0B    13    0688  1152         BEQL    71$                    ; Branch if so
02010010 8F   0008'CF   D1    068A  1153         CMPL    NET$GL_SRCH_ID,#NFB$C_NDI_TAD ; Search by transformed address?
                C9    12    0693  1154         BNEQ    40$                    ; Branch if not - skip it
        00    000C'CF   D1    0695  1155 71$:    CMPL    NET$GL_OPER,#NFB$C_OP_EQL ; Using equality match?
                C2    12    069A  1156         BNEQ    40$                    ; Branch if not
        58    0014'CF   D0    069C  1157         MOVL    NET$GQ_SRCH_KEY+4,R8   ; Get desired node address
                BB    13    06A1  1158         BEQL    40$                    ; If zero, then skip
              F95A'   30    06A3  1159         BSBW    NET$LOCATE_NDI         ; Find previous NDI position
            B5 50   E9    06A6  1160         BLBC    R0,40$                 ; If not found, then report error
                      06A9  1161         ;
                      06A9  1162         ;   Determine and save the current position context away, since
                      06A9  1163         ;   the CNF entry may not exist after a SET/CLEAR if it is new
                      06A9  1164         ;   and fails to be inserted.
                      06A9  1165         ;
        0128'CF   5A    D0    06A9  1166 75$:    MOVL    R10,PTR_OLD_CNF        ; Store CNF address
          59  14 AB   D0    06AE  1167 76$:    MOVL    CNR$L_FCD_COLL(R11),R9 ; Get field i.d. for this database
                7E    D4    06B2  1168         CLRL    -(SP)                  ; Init flag to indicate alloc failure
                5E    DD    06B4  1169         PUSHL   SP                     ; Save accessible address for copy
```

```
                F947'  30   06B6  1170          BSBW    CNF$GET_FIELD                   ; Get field's value
              19 50    E9   06B9  1171          BLBC    R0,77$                          ; Br if error
51    57      0C     C1   06BC  1172          ADDL3   #12,R7,R1                       ; Compute length of storage block
                F93D'  30   06C0  1173          BSBW    NET$ALLOCATE                    ; Allocate storage to hold string
              0F 50    E9   06C3  1174          BLBC    R0,77$                          ; Br if error
   04 AE      52     D0   06C6  1175          MOVL    R2,4(SP)                        ; Save address of allocation
   50   0C A2     9E   06CA  1176          MOVAB   12(R2),R0                       ; Point to string storage area
      6E      50     D0   06CE  1177          MOVL    R0,(SP)                         ; Save real collating value pointer
60    68      57     28   06D1  1178          MOVC3   R7,(R8),(R0)                    ; Copy string text into buffer
              57     DD   06D5  1179 77$:     PUSHL   R7                              ; Save collating length
                          06D7  1180          ;
                          06D7  1181          ;   Call action routine to process CNF fields.
                          06D7  1182          ;
            EC'AF    9F   06D7  1183          PUSHAB  B^80$                           ; Setup return address
                          06DA  1184          $DISPATCH  NFB$B_FCT(R6),TYPE=B,-       ; Dispatch on Funtion code
                          06DA  1185          <-
                          06DA  1186              <NFB$C_FC_SET,      ACTION_SET>,    -;
                          06DA  1187              <NFB$C_FC_SHOW,     ACTION_SHOW>,   -;
                          06DA  1188              <NFB$C_FC_CLEAR,    ACTION_CLEAR>,  -;
                          06DA  1189              <NFB$C_FC_DELETE,   ACTION_DELETE>, -;
                          06DA  1190              <NFB$C_FC_ZERCOU,   ACTION_ZERCOU>, -;
                          06uA  1191          >
                          06E8  1192          BUG_CHECK NETNOSTATE,FATAL
                          06EC  1193
      57    8E    7D    06EC  1194 80$:     MOVQ    (SP)+,R7                        ; Recover collating descriptor
                52 8ED0   06EF  1195          POPL    R2                              ; Restore address of allocated block
                05   13   06F2  1196          BEQL    82$                             ; If EQL, allocation failure
0000'DF       62   0E   06F4  1197          INSQUE  (R2),@NET$GQ_TMP_BUF            ; Insert onto temporary buffer queue
56    0048'CF     D0   06F9  1198 82$:     MOVL    NET$GL_PTR_PT,R6                ; Recover pointer to NFB
                          06FE  1199          ;
                          06FE  1200          ;   If operation was successful, then update the P2 context area
                          06FE  1201          ;   with the current position in the database, so that subsequent
                          06FE  1202          ;   QIOs will continue from this point.
                          06FE  1203          ;
0000'8F       50   B1   06FE  1204          CMPW    R0,#SS$_RESULTOVF               ; Result overflow?
                2E   13   0703  1205          BEQL    200$                            ; If so, don't treat as a "real error"
                00   E0   0705  1206          BBS     #NFB$V_ERRUPD,-                 ; If set, then update even on error
     03 01 A6                 0707  1207                      NFB$B_FLAGS(R6),85$
        26 50    E9   070A  1208          BLBC    R0,200$                         ; Else, if error, then don't update P2
                02   E0   070D  1209 85$:     BBS     #NFB$V_NOCTX,-                  ; If NOCTX flag set, then user wants to
     13 01 A6                 070F  1210                      NFB$B_FLAGS(R6),90$     ; stay on this entry for a while
                50   DD   0712  1211          PUSHL   R0                              ; Save final status
51    0040'CF     D0   0714  1212          MOVL    NET$GL_PTR_P2,R1                ; Point to P2 context area
      81    57    B0   0719  1213          MOVW    R7,(R1)+                        ; Enter count of bytes in string
00    68    57    2C   071C  1214          MOVC5   R7,(R8),#0,-                    ; Enter string text
      61    20         0720  1215                      #NFB$C_CTX_SIZE,(R1)
              50 8ED0   0722  1216          POPL    R0                              ; Restore final status
                          0725  1217 90$:     ;
                          0725  1218          ;   Update the CNF count and the P3 count of P4 buffer bytes used
                          0725  1219          ;
      0124'DF     D6   0725  1220          INCL    @PTR_CNFCNT                     ; Update number of complete CNF blocks
                          0729  1221                                                  ; processed
      0030'CF     A3   0729  1222          SUBW3   NET$GL_PTR_P4,-                 ; Update count of bytes used in the P4
      011C'CF              072D  1223                  PTR_L_P4,-                  ; buffer
      0038'DF              0730  1224                  @NET$GL_PTR_P3
                05   0733  1225 200$:     RSB
```

```
                     0734  1227
                     0734  1228                    .ENABL  LSB
                     0734  1229
                     0734  1230 ACTION_SET:                                ; ACP Control 'set' QIO action routine
                     0734  1231          SETBIT  NET$V_SETQIO,NET$GL_FLAGS ; Set flag to indicate QIO type
          0A   11    0739  1232          BRB     50$                       ; Continue in common
                     073B  1233
                     073B  1234 ACTION_CLEAR:                              ; ACP 'clear' QIO action routine
          02   E1    073B  1235          BBC     #CNF$V_FLG_ACP,-          ; If BS then block is a "phantom"
      05 0B AA       073D  1236                  CNF$B_FLG(R10),50$        ;
                     0740  1237          ;
                     0740  1238          ;
                     0740  1239          ;   The "phantom" CNF is being used to represent a specific database
                     0740  1240          ;   entry.  Go thru the motions of clearing the specified parameters in
                     0740  1241          ;   order detect errors (such as clearing a read-only parameter) so
                     074C  1242          ;   that this entry has the same behavior as the CNFs that exist in the
                     0740  1243          ;   database as "actual" CNF blocks.
                     0740  1244          ;
                     0740  1245          ;
        0010   30    0740  1246          BSBW    SETCLEAR                  ; Clear specified parameters
          0D   11    0743  1247          BRB     100$                      ; Delete the "new" CNF
                     0745  1248          ;
                     0745  1249          ;
                     0745  1250          ;   Attempt to SET/CLEAR the new CNF values.  If successful then
                     0745  1251          ;   attempt to replace the old CNF entry with the new one.
                     0745  1252          ;
                     0745  1253          ;
          0C   10    0745  1254 50$:     BSBB    SETCLEAR                  ; SET/CLEAR the new values
        08 50  E9    0747  1255          BLBC    R0,100$                   ; If LBC then error
    56  0128'CF  D0   074A  1256          MOVL    PTR_OLD_CNF,R6            ; Get pointer to original CNF
          F8AE'  30    074F  1257          BSBW    CNF$INSERT                ; R6 -> old,  R10 -> util on input
                     0752  1258                                            ; R10 -> whatever one makes it, R6
                     0752  1259                                            ; and original R10 are lost
                     0752  1260                                            ; Attempt to insert new CNF entry
    05               0752  1261 100$:    RSB                               ; Else return error
                     0753  1262
                     0753  1263                    .DSABL  LSB
```

```
                     0753  1265
                     0753  1266 SETCLEAR:                                ; Common SET/CLEAR processing
                     0753  1267          ;
                     0753  1268          ;   R11 = CNR pointer
                     0753  1269          ;   R10 = CNF pointer
                     0753  1270          ;   R6  = NFB pointer
                     0753  1271          ;
                     0753  1272
                     0753  1273 10$:     ;
                     0753  1274          ;
                     0753  1275          ;   See if the CNF is "locked", that is, if its conditionally
                     0753  1276          ;   writeable fields are locked and cannot be written.
                     0753  1277          ;
                     0753  1278          ;
59   10 AB    D0     0753  1279          MOVL    CNR$L_FLD_LOCK(R11),R9  ; Get i.d. of "lock" field
                     0757  1280          CLRBIT  NET$V_CNFLCK,-          ; Assume that conditionally writeable
                     0757  1281                  NET$GL_FLAGS            ; fields are writeable
      F8A0'  30      075D  1282          BSBW    CNF$GET_FIELD           ; See if it's set
    06 58    E9      0760  1283          BLBC    R8,20$                  ; If LBC then not set, not "locked"
                     0763  1284          SETBIT  NET$V_CNFLCK,-          ; Indicate that conditionally writeable
                     0763  1285                  NET$GL_FLAGS            ; fields are not writeable
                     0769  1286 20$:     ;
                     0769  1287          ;
                     0769  1288          ;   We cannot alter the only copy of the current CNF in case the Qio
                     0769  1289          ;   eventually fails.  We must create a clone and modify it.  If all
                     0769  1290          ;   goes well it will eventually replace the original CNF in the
                     0769  1291          ;   database.
                     0769  1292          ;
                     0769  1293          ;
58   0128'CF    D0     0769  1294          MOVL    PTR_OLD_CNF,R8          ; Recover pointer to "old" CNF
        09     13     076E  1295          BEQL    25$                     ; If EQL then none, R10 points to
                     0770  1296                                          ; the utility buffer already
      F88D'  30      0770  1297          BSBW    CNF$INIT_UTL            ; Init "utility buffer" as a CNF
      F88A'  30      0773  1298          BSBW    CNF$COPY                ; Copy R8 CNF to R10 CNF
    1B 50    E9      0776  1299          BLBC    R0,40$                  ; If LBC then error
                     0779  1300 25$:     ;
                     0779  1301          ;
                     0779  1302          ;   Zip down the field i.d. list in the P1 buffer.  For each field
                     0779  1303          ;   attempt to either clear or set the field according to the type of
                     0779  1304          ;   Qio being processed.
                     0779  1305          ;
                     0779  1306          ;   Before setting/clearing the field, read it so that it may be
                     0779  1307          ;   compared to the value which the Qio is trying to set (comparison
                     0779  1308          ;   for the CLEAR Qio is 'is it already clear?'; comparison for the
                     0779  1309          ;   SET Qio is 'does it already have this value').  This is done for
                     0779  1310          ;   the following reasons:
                     0779  1311          ;
                     0779  1312          ;    o  If the field is write-locked and the new value equals the old
                     0779  1313          ;       value then no error should be returned.  This is easier to
                     0779  1314          ;       check before the modification is attempted than after it fails.
                     0779  1315          ;    o  If the values are the same then the modification is not needed
                     0779  1316          ;       and the "put field" is more expensive than a "read field".
                     0779  1317          ;       Setting a field to its original value is actually too uncommon
                     0779  1318          ;       since (in NCP terms) the safest way to update both the
                     0779  1319          ;       disk resident and NETACP resident databases is with the
                     0779  1320          ;       NCP commands:
                     0779  1321          ;
```

```
                        0779  1322          ;               NCP>DEF  entity-type  entity-id  parameter
                        0779  1323          ;               NCP>SET  entity-type  entity-id  ALL
                        0779  1324          ;
                        0779  1325          ;
          55   10 A6    9E  0779  1326          MOVAB   NFB$L_FLDID(R6),R5      ; Point to the first field i.d.
59  0044'CF   0048'CF    C1  077D  1327 30$:     ADDL3   NET$GL_PTR_P1,NET$GL_SIZ_P1,R9 ; Address of end of NFB
              59   55     D1  0785  1328          CMPL    R5,R9                   ; Are we at the end of the NFB?
                     0A   1E  0788  1329          BGEQU   40$                     ; If so, then we're done
              59   85     D0  078A  1330          MOVL    (R5)+,R9                ; Get next field i.d.
                           078D  1331          ASSUME  NFB$C_ENDOFLIST  EQ  0
                   05   13  078D  1332          BEQL    40$                     ; If EQL then no more field i.d.s
                   12   10  078F  1333          BSBB    100$                    ; SET/CLEAR the field
                E9 50   E8  0791  1334          BLBS    R0,30$                  ; Loop unless error is signalled
    011C'CF   0030'CF    D0  0794  1335 40$:     MOVL    NET$GL_PTR_P4,PTR_L_P4  ; Reset the P4 descriptor for the next pass
    0118'CF   002C'CF    D0  079B  1336          MOVL    NET$GL_SIZ_P4,SIZ_L_P4
                         05  07A2  1337          RSB                             ; Return with status in R0 and error
                             07A3  1338                                          ; qualifier in R9
                             07A3  1339
                             07A3  1340 100$:    ;
                             07A3  1341          ;
                             07A3  1342          ;   If this is a SET Qio then branch.  Else, this is a CLEAR Qio --
                             07A3  1343          ;   if LBC in R0 then the field is already clear in the new CNF and
                             07A3  1344          ;   there's no need to attempt to clear it again.
                             07A3  1345          ;
                             07A3  1346          ;
              F85A'      30  07A3  1347          BSBW    CNF$GET_FLD_EX          ; Get the current field value for later
                             07A6  1348                                          ; reference using access rights of user
          013C'CF   50   B0  07A6  1349          MOVW    R0,GET_W_STATUS         ; Save status
                    00   E0  07AB  1350          BBS     #NET$V_SETQIO,-         ; If BS then SET Qio
            0C 0000'CF         07AD  1351                  NET$GL_FLAGS,105$   ;
                 06 50   E9  07B1  1352          BLBC    R0,102$                 ; If LBC then field is already clear
                  F849'      30  07B4  1353          BSBW    CNF$CLR_FLD_EX          ; Clear the field according to the
                             07B7  1354                                          ; user's access rights
                   0088      31  07B7  1355          BRW     330$                    ; Return with status in R0
                   0082      31  07BA  1356 102$:     BRW     320$                    ; Return with success in R0
                             07BD  1357 105$:    ;
                             07BD  1358          ;
                             07BD  1359          ;   This is a "SET" Qio.  If the field value is not null and it is
                             07BD  1360          ;   different than the current value in the CNF then store it into the
                             07BD  1361          ;   CNF.
                             07BD  1362          ;
                             07BD  1363          ;
          53   57    7D      07BD  1364          MOVQ    R7,R3                   ; Save the field/descriptor
58  011C'CF    D0            07C0  1365          MOVL    PTR_L_P4,R8             ; Get new parameter pointer
50  0000'8F    3C            07C5  1366          MOVZWL  #SS$_RESULTOVF,R0       ; Assume P4 is too small
              10   EF        07CA  1367          EXTZV   #NFB$V_TYP,-            ;
51  59   02                  07CC  1368                  #NFB$S_TYP,R9,R1        ; Get field type
                             07CF  1369          $DISPATCH  R1,-                 ; Dispatch on field type
                             07CF  1370          <-
                             07CF  1371              <NFB$C_TYP_V,       200$>,  -; Bit
                             07CF  1372              <NFB$C_TYP_L,       200$>,  -; Longword
                             07CF  1373              <NFB$C_TYP_S,       300$>,  -; String
                             07CF  1374          >
                             07D9  1375          BUG_CHECK       NETNOSTATE,FATAL
                             07DD  1376 200$:    ;
                             07DD  1377          ;
                             07DD  1378          ;   SET "bit" or "longword" field value
```

```
                     07DD  1379          ;
                     07DD  1380          ;
   0118'CF    04     C2  07DD  1381          SUBL    #4,SIZ_L_P4             ; Account for field size
              5E     19  07E2  1382          BLSS    330$                    ; If LSS the P4 buffer is too small
011C'CF    04 A8     9E  07E4  1383          MOVAB   4(R8),PTR_L_P4          ; Update to next parameter pointer
         58   68     D0  07EA  1384          MOVL    (R8),R8                 ; Get parameter value
              50     19  07ED  1385          BLSS    320$                    ; If LSS then treat as a NOP
      44 013C'CF     E9  07EF  1386          BLBC    GET_W_STATUS,317$       ; If LBC then param not yet set
         54   58     D1  07F4  1387          CMPL    R8,R4                   ; Does old value EQL new value ?
              3D     11  07F7  1388          BRB     315$                    ; Continue in common
                     07F9  1389 300$:        ;
                     07F9  1390          ;
                     07F9  1391          ;   SET "string" value
                     07F9  1392          ;
                     07F9  1393          ;
   0118'CF    02     C2  07F9  1394          SUBL    #2,SIZ_L_P4             ; Account for string count field
              42     19  07FE  1395          BLSS    330$                    ; If LSS then too small, report error
         57   88     3C  0800  1396          MOVZWL  (R8)+,R7                ; Get string size
         52   57     D0  0803  1397          MOVL    R7,R2                   ; Make a copy
      51   0E A6     3C  0806  1398          MOVZWL  NFB$W_CELL_SIZE(R6),R1  ; Get fixed string cell size
              0B     13  080A  1399          BEQL    310$                    ; If EQL then cell size is not fixed
         51   02     A2  080C  1400          SUBW    #2,R1                   ; Adjust for count field
         52   51     D0  080F  1401          MOVL    R1,R2                   ; Set amount of P4 space used by cell
         57   51     B1  0812  1402          CMPW    R1,R7                   ; Is string size bigger than cell?
              2B     1F  0815  1403          BLSSU   330$                    ; If LSS then signal the error
011C'CF    6842     9E  0817  1404 310$:    MOVAB   (R8)[R2],PTR_L_P4       ; Store address of next field
  0118'CF    52     C2  081D  1405          SUBL    R2,SIZ_L_P4             ; Calculate P4 buffer bytes remaining
              1E     19  0822  1406          BLSS    330$                    ; If LSS then P4 buffer is too small
              57     D5  0824  1407          TSTL    R7                      ; Is the string null?
              17     13  0826  1408          BEQL    320$                    ; If EQL yes, treat as a NOP
   0B 013C'CF        E9  0828  1409          BLBC    GET_W_STATUS,317$       ; If LBC then param not yet set
         57   53     D1  082D  1410          CMPL    R3,R7                   ; Are old and new strings of equal size
              06     12  0830  1411          BNEQ    317$                    ; If NEQ then must set new value
68    64     53     29  0832  1412          CMPC3   R3,(R4),(R8)            ; Is old value EQL new value
              07     13  0836  1413 315$:    BEQL    320$                    ; If EQL then no need for set
              50     D4  0838  1414 317$:    CLRL    R0                      ; No pre-set error code
          F7C3'      30  083A  1415          BSBW    CNF$PUT_FLD_EX          ; Attempt to store new value
              03     11  083D  1416          BRB     330$                    ; Take common exit with status in R0
         50   01     D0  083F  1417 320$:    MOVL    #1,R0                   ; Indicate success
                     05  0842  1418 330$:    RSB
```

```
                     0843  1420 ACTION_DELETE:                          ; ACP 'Delete' QIO action routine
                     0843  1421          SETBIT  NET$V_DELETE,NET$GL_FLAGS ; Indicate function type
                     0848  1422          ;
                     0848  1423          ;   First move the specified fields to the P4 buffer if it exists
                     0848  1424          ;
           50   01  D0  0848  1425          MOVL    #1,R0                    ; Assume success
     002C'CF    D5  0848  1426          TSTL    NET$GL_SIZ_P4            ; Is there a P4 buffer?
           05   13  084F  1427          BEQL    10$                      ; If EQL no, continue
           0C   10  0851  1428          BSBB    ACTION_SHOW              ; Move the fields to the P4 buffer
        03 50   E9  0853  1429          BLBC    R0,20$                   ; If LBC then error
                     0856  1430          ;
                     0856  1431          ;
                     0856  1432          ;   Mark the CNF for deletion.
                     0856  1433          ;
                     0856  1434          ;
         F7A7'   30  0856  1435 10$:     BSBW    CNF$DELETE               ; Attempt to mark CNF for delete
           05   05  0859  1436 20$:     RSB                              ; Return status in R0, qualifier in R9
                     035A  1437
                     085A  1438
                     085A  1439 ACTION_ZERCOU:                           ; Zero and optionally read counters
                     085A  1440          SETBIT  NET$V_CLRCNT,NET$GL_FLAGS ; Flag "clear counters"
                     085F  1441                                           ; and fall thru
                     085F  1442
                     085F  1443
                     085F  1444 ACTION_SHOW:                             ; "SHOW" Qio action routine
         F79E'   30  085F  1445          BSBW    CNF$PRE_SHOW             ; Pre-process the CNF for "show" QIO
        3B 50   E9  0862  1446          BLBC    R0,40$                   ; Branch if error detected
                     0865  1447          ;
                     0865  1448          ;
                     0865  1449          ;   Move each field specified in the NFB into the P4 buffer.
                     0865  1450          ;
                     0865  1451          ;
              55   10 A6   9E  0865  1452          MOVAB   NFB$L_FLDID(R6),R5       ; Get address of first field i.d.
     0120'CF   011C'CF   D0  0869  1453          MOVL    PTR_L_P4,PTR_L_OLDP4     ; Save current position in P4
59   0044'CF   0048'CF   C1  0870  1454 20$:     ADDL3   NET$GL_PTR_PT,NET$GL_SIZ_P1,R9 ; Address of end of NFB
                59   55   D1  0878  1455          CMPL    R5,R9                    ; Are we at the end of the NFB?
                     20   1E  087B  1456          BGEQU   30$                      ; If so, then we're done
                59   85   D0  087D  1457          MOVL    (R5)+,R9                 ; Get next field i.d.
                     0880  1458          ASSUME  NFB$C_ENDOFLIST EQ 0
                     1B   13  0880  1459          BEQL    30$                      ; If ENDOFLIST, then we're done
           53   011C'CF   D0  0882  1460          MOVL    PTR_L_P4,R3              ; Get pointer into P4 buffer
                     21   10  0887  1461          BSBB    100$                     ; Dispatch on field type
                  15 50   E9  0889  1462          BLBC    R0,50$                   ; If LBC then error
           011C'CF   53   D0  088C  1463          MOVL    R3,PTR_L_P4              ; Update pointer into P4 buffer
           50   0000'DF   0F  0891  1464 2?$:     REMQUE  @NET$GQ_TMP_BUF,R0       ; Drain the temp buffer queue to keep
                     D8   1D  0896  1465          BVS     20$                      ; The pool as available as possible
                     F765'   30  0898  1466          BSBW    NET$DEALLOCATE           ; (CNF$GET_FIELD may have allocated one)
                     F4   11  089B  1467          BRB     25$                      ; Drain the entire queue
                     0000  1468                                           ; Then loop on each field
                50   01   D0  089D  1469 30$:     MOVL    #1,R0                    ; Indicate success
                     05   05  08A0  1470 40$:     RSB                              ; Done
                     0000  1471          ;
                     0000  1472          ; Don't return partial node entries
                     0000  1473          ;
     011C'CF   0120'CF   D0  08A1  1474 50$:     MOVL    PTR_L_OLDP4,PTR_L_P4     ; Copy old P4 pointer
                     F6   11  08A8  1475          BRB     40$                      ; And leave
                     08AA  1476
```

```
                F753'  30   08AA  1477 100$:   BSBW    CNF$GET_FLD_EX          ; Get the field/descriptor and possibly
                       08AD  1478                                              ; zero counters as a side effect
                  02   E1   08AD  1479          BBC     #NET$V_CLRCNT,-         ; If BC not ZERO COUNTER function
          06 0000'CF         08AF  1480                  NET$GL_FLAGS,105$       ;
             002C'CF   D5   08B3  1481          TSTL    NET$GL_SIZ_P4           ; Is there a user P4 buffer ?
                  61   18   08B7  1482          BGEQ    200$                    ; If GEQ no, not a READ-and-ZERO
                  10   EF   08B9  1483 105$:    EXTZV   #NFB$V_TYP,-            ;
        51   59   02         08BB  1484                  #NFB$S_TYP,R9,R1        ; Get field type
                               08BE  1485          $DISPATCH  R1,-              ; Dispatch on field type
                               08BE  1486          <-
                               08BE  1487              <NFB$C_TYP_V,       110$>,  -;
                               08BE  1488              <NFB$C_TYP_L,       110$>,  -;
                               08BE  1489              <NFB$C_TYP_S,       140$>,  -;
                               08BE  1490          >
                               08C8  1491          BUG_CHECK       NETNOSTATE,FATAL
                               08CC  1492 110$:    ;
                               08CC  1493          ;
                               08CC  1494          ;   The field is not a "string".  If the field is valid then store it
                               08CC  1495          ;   into the P4 buffer.  Else store the value -1.
                               08CC  1496          ;
                               08CC  1497          ;
              03   50   E8   08CC  1498          BLBS    R0,120$                 ; If LBS then field is valid
                 58   01   CE   08CF  1499          MNEGL   #1,R8                   ; Else use -1
          0118'CF   04   C2   08D2  1500 120$:    SUBL    #4,SIZ_L_P4             ; Account for bytes to be taken
                  45   19   08D7  1501          BLSS    220$                    ; If LSS then P4 is too small
              83   58   D0   08D9  1502          MOVL    R8,(R3)+                ; Move field value to P4 buffer
                  3C   11   08DC  1503          BRB     200$                    ; Take common exit
                               08DE  1504 140$:    ;
                               08DE  1505          ;   The field is type "string".  If field is valid then store it into
                               08DE  1506          ;   the P4 buffer.  Else store a null string.
                               08DE  1507          ;
                    05   50   E8   08DE  1508          BLBS    R0,150$                 ; If LBS then field is valid
                         57   D4   08E1  1509          CLRL    R7                      ; Nullify count if type string
                      58   5E   D0   08E3  1510          MOVL    SP,R8                   ; Point R8 to somewhere accessible
                               08E6  1511          ;
                               08E6  1512          ;       Do not return half filled parameter!
                               08E6  1513          ;
                59   0118'CF   D0   08E6  1514 150$:    MOVL    SIZ_L_P4,R9             ; Get size of P4 buffer
                     59   02   C2   08EB  1515          SUBL    #2,R9                   ; Account for bytes to be taken
                          2E   19   08EE  1516          BLSS    220$                    ; If LSS then P4 is too small
                     83   57   B0   08F0  1517          MOVW    R7,(R3)+                ; Enter count field
                     50   57   D0   08F3  1518          MOVL    R7,R0                   ; Assume string size = space used
                   51   0E   A6   3C   08F6  1519          MOVZWL  NFB$W_CELL_SIZE(R6),R1  ; Get fixed cell size
                          09   13   08FA  1520          BEQL    160$                    ; If EQL then cell size is not fixed
                50   51   02   C3   08FC  1521          SUBL3   #2,R1,R0                ; Compute space used by cell
                     50   57   D1   0900  1522          CMPL    R7,R0                   ; Is string bigger than cell size?
                          19   1A   0903  1523          BGTRU   220$                    ; If so, then signal an error
                     59   50   C2   0905  1524 160$:    SUBL    R0,R9                   ; Account for bytes to be taken
                          14   19   0908  1525          BLSS    220$                    ; If LSS then P4 is too small
                          55   DD   090A  1526          PUSHL   R5                      ; Save critical reg
63   50   00   68   57   2C   090C  1527          MOVC5   R7,(R8),#0,R0,(R3)      ; Move string text to cell
                          55 8ED0   0912  1528          POPL    R5                      ; Restore reg
                0118'CF   59   D0   0915  1529          MOVL    R9,SIZ_L_P4             ; Set size remaining in P4 buffer
                     50   01   D0   091A  1530 200$:    MOVL    #1,R0                   ; Indicate success
                          05   091D  1531          RSB
                               091E  1532
                50   0000'8F   3C   091E  1533 220$:    MOVZWL  #SS$_RESULTOVF,R0       ; Indicate P4 or cell is too small
```

```
 05  0923  1534          RSB
     0924  1535
     0924  1536
```

```
                     0924  1538 ;+
                     0924  1539 ; REISSUE_X25   - Reissue X25 QIO
                     0924  1540 ;
                     0924  1541 ; The IO$_ACPCONTROL QIO is reissued to the X25 ACP since the database
                     0924  1542 ; addressed by the QIO is maintained by that ACP.  If there is no channel
                     0924  1543 ; currently active to the X25 ACP then one is assigned.
                     0924  1544 ;
                     0924  1545 ;
                     0924  1546 ;-
                     0924  1547 REISSUE_X25:                                ; Re-issue QIO to X25 ACP
 0160'CF    B5       0924  1548          TSTW    NET$GW_X25_CHAN            ; Is there an active channel?
      05    12       0928  1549          BNEQ    50$                        ; If NEQL then yes
      43    10       092A  1550          BSBB    NET$GET_X25_CHAN           ; Assign channel, get PSI mutex
   3F 50    E9       092C  1551          BLBC    R0,100$                    ; If LBC then error
                     092F  1552 50$:     $QIOW_S FUNC = #IO$_ACPCONTROL     -; Re-issue QIO
                     092F  1553                  EFN  = #NET$C_EFN_WAIT     -; event flag for synchronous calls
                     092F  1554                  CHAN = NET$GW_X25_CHAN     -;
                     092F  1555                  IOSB = QUAD_BUF            -; Scratch quadword buffer
                     092F  1556                  P1   = NET$GL_SIZ_P1       -; Address of NFB descriptor
                     092F  1557                  P2   = #NET$GL_SIZ_P2      -; Address of P2 buffer desciptor
                     092F  1558                  P3   = NET$GL_PTR_P3       -; Address of word to return P4 bytecnt
                     092F  1559                  P4   = #NET$GL_SIZ_P4       ; Address of P4 buffer
      0D 50    E9       095E  1560          BLBC    R0,100$                    ; If LBC then error
50    0140'CF    7D       0961  1561          MOVQ    QUAD_BUF,R0                ; Setup IOSB image
      05 50    E8       0966  1562          BLBS    R0,100$                    ; Branch if successful
0004'CF    51    D0       0969  1563          MOVL    R1,NET$GQ_USR_STAT+4       ; Store error qualifier in IOSB
                  05       096E  1564 100$:    RSB                        ; Done
```

```
                        096F  1566 ;+
                        096F  1567 ; NET$GET_X25_CHAN      - Assign channel to the PSIACP and get its mutex
                        096F  1568 ;
                        096F  1569 ; A channel is assigned to the NW device.  This is the path to the PSI ACP.
                        096F  1570 ; If successful, then issue a $QIO to obtain the PSI ACP database mutex.
                        096F  1571 ; If that fails then deassign the channel.
                        096F  1572 ;
                        096F  1573 ;
                        096F  1574 ; INPUTS:        None
                        096F  1575 ;
                        096F  1576 ; OUTPUTS:       R0      Status
                        096F  1577 ;
                        096F  1578 ;-
                        096F  1579 NET$GET_X25_CHAN::                       ; Get channel to X25 ACP
                        096F  1580                  ;
                        096F  1581                  ;   ASSIGN a channel to the NW driver.  This is the path to the
                        096F  1582                  ;   PSI ACP.  The only expected error return if SS$_NOSUCHDEV
                        096F  1583                  ;   indicating that the NW driver has not been loaded.
                        096F  1584                  ;
                        096F  1585                  $ASSIGN_S -                     ; Assign channel to X25 ACP
                        096F  1586                          CHAN    = NET$GW_X25_CHAN,-
                        096F  1587                          DEVNAM = NET$GQ_X25_DEV,-
                        096F  1588                          MBXNAM = NET$GQ_MBX_NAME
                 46 50   E9 0984  1589                  BLBC    R0,200$                 ; If LBC then X25 is not active
                        0987  1590                  ;
                        0987  1591                  ;   NETACP is to be the sole modifier of the PSIACP database (other
                        0987  1592                  ;   processes to issue $QIO's to show the PSIACP database).  Thus, a
                        0987  1593                  ;   $QIO must be issued to obtain the PSIACP database mutex.
                        0987  1594                  ;
                        0987  1595                  ;   The  expected return status codes are:
                        0987  1596                  ;
                        0987  1597                  ;       SS$_NORMAL      if successful
                        0987  1598                  ;       SS$_DEVACTIVE   if the mutex is already owned
                        0987  1599                  ;       SS$_NOSUCHDEV   if the PSIACP is not yet running
                        0987  1600                  ;
                        0987  1601                  $QIOW_S EFN  = #NET$C_EFN_WAIT,-; Event flag for synchronous calls
                        0987  1602                          IOSB = QUAD_BUF,-       ; Scratch quadword buffer
                        0987  1603                          CHAN = NET$GW_X25_CHAN,-;
                        0987  1604                          FUNC = #IO$_INITIALIZE!IO$M_ACCESS ; Ask for the mutex
                     0D 50   E9 09A8  1605                  BLBC    R0,100$                 ; If LBC then error
               50  0140'CF   7D 09AB  1606                  MOVQ    QUAD_BUF,R0             ; Setup IOSB image
                     1A 50   E8 09B0  1607                  BLBS    R0,200$                 ; If LBS then no error
          0004'CF    51   D0 09B3  1608                  MOVL    R1,NET$GQ_USR_STAT+4    ; Set error qualifier in IOSB
                        09B8  1609                  ;
                        09B8  1610                  ;   The attempt to obtain the mutex has failed.  $DASSGN the channel in
                        09B8  1611                  ;   order to leave our database consistent, and it order to allow the
                        09B8  1612                  ;   PSIACP to assign a channel to the one and only NW UCB (the template
                        09B8  1613                  ;   bit is set to allow NW UCBs to be cloned after PSIACP initializes).
                        09B8  1614                  ;
                     50   DD 09B8  1615 100$:      PUSHL   R0                      ; Save error status
                        09BA  1616                  $DASSGN_S NET$GW_X25_CHAN       ; Deassign the channel
                0160'CF   B4 09C6  1617                  CLRW    NET$GW_X25_CHAN         ; Zero indicates "no channel assigned"
                     50 8ED0 09CA  1618                  POPL    R0                      ; Restore original status
                        05   09CD  1619 200$:      RSB                             ; Done
                        09CE  1620
                        09CE  1621
                        09CE  1622 .END
```

```
$$T1                    = 00000001
ABD$C_FIB               = 00000001
ABD$C_LENGTH            = 00000008
ABD$C_NAME              = 00000002
ABD$C_RES               = 00000004
ABD$C_WINDOW            = 00000000
ABD$W_COUNT             = 00000002
ABD$W_TEXT              = 00000000
ACP$C_STA_F             = 00000004
ACP$C_STA_H             = 00000005
ACP$C_STA_I             = 00000000
ACP$C_STA_N             = 00000001
ACP$C_STA_R             = 00000002
ACP$C_STA_S             = 00000003
ACTION_CLEAR              0000073B R      04
ACTION_DELETE             00000843 R      04
ACTION_SET                00000734 R      04
ACTION_SHOW               0000085F R      04
ACTION_ZERCOU             0000085A R      04
BADPARAM1                 0000022F R      04
BAD_PARAM                 00000498 R      04
BIT...                  = 00000006
BUG$_NETNOSTATE           ********   X    04
CANCEL_COMMON             0000033C R      04
CANCEL_L_PID              0000016E R      02
CANCEL_W_CHN              0000017E R      02
CNF$B_FLG               = 0000000B
CNF$CLR_FIELD             ********   X    04
CNF$CLR_FLD_EX            ********   X    04
CNF$COPY                  ********   X    04
CNF$DELETE                ********   X    04
CNF$GET_FIELD             ********   X    04
CNF$GET_FLD_EX            ********   X    04
CNF$INIT_UTL              ********   X    04
CNF$INSERT                ********   X    04
CNF$KEY_SEARCH            ********   X    04
CNF$KEY_SRCH_EX           ********   X    04
CNF$PRE_QIO               ********   X    04
CNF$PRE_SHOW              ********   X    04
CNF$PURGE                 ********   X    04
CNF$PUT_FIELD             ********   X    04
CNF$PUT_FLD_EX            ********   X    04
CNF$SEARCH                ********   X    04
CNF$SEARCH_EX             ********   X    04
CNF$VERIFY                ********   X    04
CNF$V_FLG_ACP           = 00000002
CNF$_ADVANCE            = 00000000
CNF$_QUIT               = 00000002
CNF$_TAKE_CURR          = 00000003
CNF$_TAKE_PREV          = 00000001
CNRSL_FLD_COLL          = 00000014
CNRSL_FLD_LOCK          = 00000010
COMSPOST                  ********   X    04
CREATE_OBI                00000233 R      04
CTL_DATABASE              000003E4 R      04
CTL_DCLZNA                00000150 R      02
CTL_Q_DCLZNA              00000148 R      02
DCL_COMMON                0000019A R      04
DCL_NAME                  00000184 R      04
DCL_OBJECT                00000147 R      04
DCL_SERVER                00000275 R      04
DISPATCH                  000000CD R      04
DUMMY_P2                  0000004C R      02
DUMMY_P2_LNG            = 000000C8
DUMMY_P3                  00000114 R      02
DUMMY_P4                  0000004C R      02
DUMMY_P4_LNG            = 000000C8
EXE$IPID_TO_EPID          ********     X  04
GET_P2_KEY                00000597 R      04
GET_W_STATUS              0000013C R      02
ILLFCT                    0000013C R      04
ILL_FUNC                  0000048D R      04
IO$M_ACCESS               ********     X  04
IO$_ACPCONTROL            ********     X  04
IO$_INITIALIZE            ********     X  04
IRP$L_IOST1             = 00000038
IRP$L_PID               = 0000000C
IRP$L_SVAPTE            = 0000002C
IRP$L_UCB               = 0000001C
IRP$Q_NT_PRVMSK         = 00000040
IRP$V_FUNC              = 00000001
IRP$W_CHAN              = 00000028
IRP$W_STS               = 0000002A
LOCAL_L_FLAG              0000012C R      02
NET$ACP_CANCEL            0000032B RG     04
NET$ALLOCATE              ********     X  04
NET$AL_CNR_TAB            ********     X  04
NET$AL_SRCH_LIST          00000008 R      02
NET$BIN2ASC               ********     X  04
NET$CONTROL_QIO           00000000 RG     04
NET$C_ACT_TIMER         = 0000001E
NET$C_DR_EXIT           = 00000026
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  04
```

```
NET$DEC_TRANS                   ********    X   04
NET$DRV_CANCEL                  0000031F RG     04
NET$GETOTLBUF                   ********    X   04
NET$GET_X25_CHAN                0000096F RG     04
NET$GL_CNR_OBI                  ********    X   04
NET$GL_CNR_SPI                  ********    X   04
NET$GL_FLAGS                    ********    X   04
NET$GL_OPER                     0000000C R      02
NET$GL_OPER2                    0000001C R      02
NET$GL_PM_IN                    00000004 R      02
NET$GL_PM_OUT                   00000000 R      02
NET$GL_PTR_P1                   00000048 RG     02
NET$GL_PTR_P2                   00000040 RG     02
NET$GL_PTR_P3                   00000038 RG     02
NET$GL_PTR_P4                   00000030 RG     02
NET$GL_PTR_VCB                  ********    X   04
NET$GL_SAVE_IRP                 ********    X   04
NET$GL_SAVE_UCB                 ********    X   04
NET$GL_SIZ_P1                   00000044 RG     02
NET$GL_SIZ_P2                   0000003C RG     02
NET$GL_SIZ_P3                   00000034 RG     02
NET$GL_SIZ_P4                   0000002C RG     02
NET$GL_SRCH2_ID                 00000018 RG     02
NET$GL_SRCH_ID                  00000008 RG     02
NET$GQ_MBX_NAME                 ********    X   04
NET$GQ_SRCH2_KEY                00000020 RG     02
NET$GQ_SRCH_KEY                 00000010 RG     02
NET$GQ_TMP_BUF                  ********    X   04
NET$GQ_USR_STAT                 ********    X   04
NET$GQ_X25_DEV                  00000004 RG     03
NET$GW_X25_CHAN                 00000160 RG     02
NET$LOCATE_NDI                  ********    X   04
NET$LOG_EVENT                   ********    X   04
NET$M_MAXLNKMSK               = 000003FF
NET$READ_EVENT                  ********    X   04
NET$RESEND_SERVER               ********    X   04
NET$SCAN_FOR_ZNA                ********    X   04
NET$SERVER_FAIL                 ********    X   04
NET$V_BYPASS                  = 00000008
NET$V_CLRCNT                  = 00000002
NET$V_CNFLCK                  = 0000000B
NET$V_DELETE                  = 00000003
NET$V_PURGE                   = 0000000E
NET$V_SETQIO                  = 00000000
NFB$B_DATABASE                = 00000002
NFB$B_FCT                     = 00000000
NFB$B_FLAGS                   = 00000001
NFB$B_MBZ1                    = 0000000D
NFB$B_OPER                    = 00000003
NFB$B_OPER2                   = 0000000C
NFB$C_CTX_SIZE                = 00000020
NFB$C_DB_MAX                  = 0000001B
NFB$C_DB_NDI                  = 00000002
NFB$C_DB_PSI1                 = 00000015
NFB$C_DB_PSI2                 = 00000016
NFB$C_DB_PSI3                 = 00000017
NFB$C_DB_PSI4                 = 00000018
NFB$C_DB_PSI5                 = 00000019
NFB$C_DB_XAI                  = 0000001B
NFB$C_DB_XD5                  = 0000000D
NFB$C_DB_XD9                  = 0000000F
NFB$C_DB_XDI                  = 0000000B
NFB$C_DB_XGI                  = 0000000A
NFB$C_DB_XNI                  = 00000009
NFB$C_DB_XS5                  = 0000000C
NFB$C_DB_XS9                  = 0000000E
NFB$C_DB_XTI                  = 00000010
NFB$C_DB_XTT                  = 00000011
NFB$C_DECLNAME                = 00000015
NFB$C_DECLOBJ                 = 00000016
NFB$C_DECLSERV                = 00000017
NFB$C_ENDOFLIST               = 00000000
NFB$C_FC_CLEAR                = 00000024
NFB$C_FC_DELETE               = 00000021
NFB$C_FC_MAX                  = 00000026
NFB$C_FC_SET                  = 00000023
NFB$C_FC_SHOW                 = 00000022
NFB$C_FC_ZERCOU               = 00000025
NFB$C_LOGEVENT                = 0000001C
NFB$C_NDI_ADD                 = 02010012
NFB$C_NDI_TAD                 = 02010010
NFB$C_OBI_CHN                 = 03010013
NFB$C_OBI_NAM                 = 03020044
NFB$C_OBI_NUM                 = 03010014
NFB$C_OBI_PID                 = 03010015
NFB$C_OBI_SET                 = 03000002
NFB$C_OBI_UCB                 = 03010012
NFB$C_OBI_ZNA                 = 03020041
NFB$C_OP_EQL                  = 00000000
NFB$C_OP_FNDPOS               = 00000006
NFB$C_OP_MAXFCT               = 00000003
NFB$C_READEVENT               = 0000001D
NFB$C_SPI_CHN                 = 12010012
NFB$C_SPI_IRP                 = 12010011
NFB$C_SPI_NCB                 = 12020044
NFB$C_SPI_PID                 = 12010010
NFB$C_SPI_PNM                 = 12020045
NFB$C_SPI_SFI                 = 12020043
NFB$C_TYP_L                   = 00000001
NFB$C_TYP_S                   = 00000002
NFB$C_TYP_STR                 = 00000002
NFB$C_TYP_V                   = 00000000
NFB$C_WILDCARD                = 00000001
NFB$L_FLDID                   = 00000010
NFB$L_SRCH2_KEY               = 00000008
NFB$L_SRCH_KEY                = 00000004
NFB$S_TYP                     = 00000002
NFB$V_ERRUPD                  = 000C0000
NFB$V_MULT                    = 00000001
NFB$V_NOCTX                   = 00000002
NFB$V_TYP                     = 00000010
NFB$W_CELL_SIZE               = 0000000E
NFB$_ERR_CELL                 = 00000009
NFB$_ERR_DB                   = 00000002
```

```
NFB$_ERR_FCT                    = 00000001
NFB$_ERR_OPER                   = 0000000A
NFB$_ERR_P1                     = 00000003
NFB$_ERR_P2                     = 00000004
NFB$_ERR_P3                     = 00000005
NFB$_ERR_SRCH                   = 0000000B
NFB$_ERR_SRCH2                  = 0000000C
NO_PRV                            00000133 R      04
NSP$C_EXT_LNK                   = 0000001E
NSP$C_MAXADR                    = 00000009
P1_ABD_CNT                        00000138 R      02
P2_ABD_CNT                        00000134 R      02
P4_ABD_CNT                        00000130 R      02
PROCESS_CNF                       000005F6 R      04
PRV$V_BYPASS                    = 0000001D
PRV$V_DIAGNOSE                  = 00000006
PRV$V_OPER                      = 00000012
PRV$V_SYSNAM                    = 00000002
PRV_Q_REQ                         00000010 R      03
PTR_CNFCNT                        00000124 R      02
PTR_L_OLDP4                       00000120 R      02
PTR_L_P4                          0000011C R      02
PTR_OLD_CNF                       00000128 R      02
QUAD_BUF                          00000140 R      02
RCB$Q_TRANS                     = 0000000C
REISSUE_X25                       00000924 R      04
SETCLEAR                          00000753 R      04
SIZ...                          = 00000001
SIZ_L_P4                          00000118 R      02
SPI_CANCEL_SRCH                   00000162 R      02
SS$_ABORT                         ********    X   04
SS$_BADPARAM                      ********    X   04
SS$_ENDOFFILE                     ********    X   04
SS$_ILLCNTRFUNC                   ********    X   04
SS$_NOMBX                         ********    X   04
SS$_NOPRIV                        ********    X   04
SS$_NORMAL                        ********    X   04
SS$_RESULTOVF                     ********    X   04
SS$_WRITLCK                       ********    X   04
SYS$ASSIGN                        ********  GX    04
SYS$DASSGN                        ********  GX    04
SYS$QIOW                          ********  GX    04
TMP                             = 00000170 R      03
TMPMASK                         = 00040000
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$C_PRI_RTHRU                  = 0000001F
UCB$C_AMB                       = 00000060
WRTBCRFCT                         00000148 R      03
X25_DB_MASK                       00000000 R      03
_$$_                            = 000000EF
```

### 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 | 00000186 ( 390.) | 02 ( 2.) | NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG |
| NET_PURE | 00000170 ( 368.) | 03 ( 3.) | NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG |
| NET_CODE | 000009CE ( 2510.) | 04 ( 4.) | NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG |

### Performance indicators

| Phase | Page faults | CPU Time | Elapsed Time |
|---|---|---|---|
| Initialization | 37 | 00:00:00.06 | 00:00:00.25 |
| Command processing | 176 | 00:00:00.97 | 00:00:05.09 |
| Pass 1 | 488 | 00:00:20.82 | 00:00:43.14 |
| Symbol table sort | 0 | 00:00:02.36 | 00:00:04.45 |
| Pass 2 | 331 | 00:00:05.27 | 00:00:10.90 |
| Symbol table output | 35 | 00:00:00.31 | 00:00:01.04 |
| Psect synopsis output | 2 | 00:00:00.03 | 00:00:00.03 |
| Cross-reference output | 0 | 00:00:00.00 | 00:00:00.00 |
| Assembler run totals | 1071 | 00:00:29.84 | 00:01:04.92 |

The working set limit was 2000 pages.
107916 bytes (211 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1492 non-local and 119 local symbols.
1622 source lines were read in Pass 1, producing 32 object records in Pass 2.
48 pages of virtual memory were used to define 44 macros.

### Macro library statistics

| Macro library name | Macros defined |
|---|---|
| _$255$DUA28:[SHRLIB]NMALIBRY.MLB;1 | 0 |
| _$255$DUA28:[SHRLIB]EVCDEF.MLB;1 | 0 |
| _$255$DUA28:[NETACP.OBJ]NETDRV.MLB;1 | 0 |
| _$255$DUA28:[NETACP.OBJ]NET.MLB;1 | 16 |
| _$255$DUA28:[SYS.OBJ]LIB.MLB;1 | 4 |
| _$255$DUA28:[SYSLIB]STARLET.MLB;2 | 14 |
| TOTALS (all libraries) | 34 |

1706 GETS were required to define 34 macros.

There were no errors, warnings or information messages.

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

NETDLE
LIS

NETCONFIG
LIS

NETCONECT
LIS

NETCTLALL
LIS