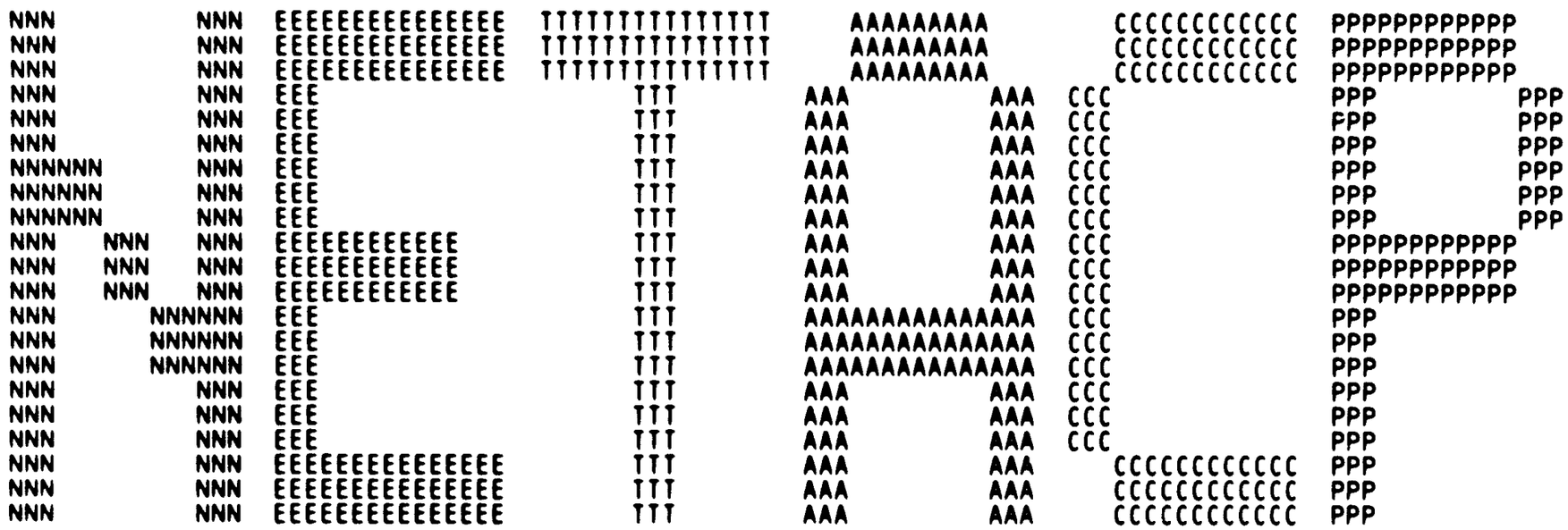
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN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

FILEID**NETCNFDLL

```
NN       NN  EEEEEEEEEE  TTTTTTTTTT    CCCCCCCC  NN       NN  FFFFFFFFFF  DDDDDDDD    LL          LL
NN       NN  EEEEEEEEEE  TTTTTTTTTT    CCCCCCCC  NN       NN  FFFFFFFFFF  DDDDDDDD    LL          LL
NN       NN  EE              TT       CC        NN       NN  FF          DD      DD  LL          LL
NN       NN  EE              TT       CC        NN       NN  FF          DD      DD  LL          LL
NNNN     NN  EE              TT       CC        NNNN     NN  FF          DD      DD  LL          LL
NNNN     NN  EE              TT       CC        NNNN     NN  FF          DD      DD  LL          LL
NN  NN   NN  EEEEEEEE        TT       CC        NN  NN   NN  FFFFFFFF    DD      DD  LL          LL
NN  NN   NN  EEEEEEEE        TT       CC        NN  NN   NN  FFFFFFFF    DD      DD  LL          LL
NN    NNNN  EE              TT       CC        NN    NNNN  FF          DD      DD  LL          LL
NN    NNNN  EE              TT       CC        NN    NNNN  FF          DD      DD  LL          LL
NN      NN  EE              TT       CC        NN      NN  FF          DD      DD  LL          LL          ....
NN      NN  EE              TT       CC        NN      NN  FF          DD      DD  LL          LL          ....
NN      NN  EEEEEEEEEE      TT         CCCCCCCC  NN      NN  FF          DDDDDDDD    LLLLLLLLLL  LLLLLLLLLL  ....
NN      NN  EEEEEEEEEE      TT         CCCCCCCC  NN      NN  FF          DDDDDDDD    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  NETCNFDLL  - Datalink database action routines
0000     2          .IDENT  'V04-000'
0000     3          .DEFAULT DISPLACEMENT,LONG
0000     4
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27  ;
0000    28
0000    29  ; FACILITY:     NETWORK ACP
0000    30  ;
0000    31  ; ABSTRACT:
0000    32  ;                Action routines for CRI and PLI databases.
0000    33  ;
0000    34  ;                This module provides support to the NETACP database management
0000    35  ;                including database entry insertion and action routines to
0000    36  ;                retrieve data for parameters which are not stored in any of
0000    37  ;                the datalink CNF control blocks (CRI and PLI).
0000    38  ;
0000    39  ; ENVIRONMENT:
0000    40  ;                The module runs in kernel mode and at possibly elevated IPL.
0000    41  ;                It is therefore locked into the ACP's virtural address space
0000    42  ;                in order to prevent the need for paging.
0000    43  ;
0000    44  ;                Since the ACP is work-queue driven, and since it is the ACP
0000    45  ;                that modifies the structure of the non-paged pool data base
0000    46  ;                including the RCB (actually a VCB) and the private structures
0000    47  ;                hanging off of the RCB, there is no need to obtain the system
0000    48  ;                data base mutex -- no races can occur.   However, it is
0000    49  ;                necessary to raise IPL in order to stop any races with
0000    50  ;                NETDRIVER.
0000    51  ;
0000    52  ;
0000    53  ; AUTHOR:       A.Eldridge      14-Feb-80
0000    54  ;
0000    55  ; MODIFIED BY:
0000    56  ;
0000    57  ;       V021    RNG0021         Rod Gamache     17-Jul-1984
```

```
0000     58 ;                  Add more checks to insure the DEV$V_NET bit is set for
0000     59 ;                  datalink drivers.
0000     60 ;
0000     61 ;          V020    RNG0020         Rod Gamache     16-Mar-1984
0000     62 ;                  Make sure DEV$V_NET bit is on before attempting to use any
0000     63 ;                  device driver.
0000     64 ;
0000     65 ;          V019    RNG0019         Rod Gamache     13-Jan-1984
0000     66 ;                  Add DPV and KMV support for X.25.
0000     67 ;
0000     68 ;          V018    RNG0018         Rod Gamache     18-Nov-1983
0000     69 ;                  Add QNA support for communication over the NI.
0000     70 ;
0000     71 ;          V017    TMH0017         Tim Halvorsen   27-Apr-1983
0000     72 ;                  Add pseudo-point-to-point version of PPUNA circuit for
0000     73 ;                  testing of point-to-point circuits over the NI.
0000     74 ;
0000     75 ;          V016    RNG0016         Rod Gamache     15-Apr-1983
0000     76 ;                  Allow driver NMA parameter i.d.'s to be returned which
0000     77 ;                  the NETACP doesn't know about. This allows drivers the
0000     78 ;                  flexibility of adding parmeters which the NETACP can ignore.
0000     79 ;
0000     80 ;          V015    RNG0015         Rod Gamache     11-Apr-1983
0000     81 ;                  Make the REMOVE_XXX routines call the REMOVE_DEF routine
0000     82 ;                  to remove the PCI or CRI CNF structure.
0000     83 ;
0000     84 ;          V014    TMH0014         Tim Halvorsen   04-Mar-1983
0000     85 ;                  Add new DEVNAM circuit parameter action routine.
0000     86 ;                  Return second longword of IOSB from datalink setmode
0000     87 ;                  to issuer of SET LINE STATE ON function.
0000     88 ;
0000     89 ;          V013    TMH0013         Tim Halvorsen   14-Feb-1983
0000     90 ;                  Make datalink BUS parameter variable, depending on the
0000     91 ;                  type of datalink.  It now returns a larger value for NI
0000     92 ;                  datalinks, to account for the additional Phase IV Ethernet
0000     93 ;                  route header.
0000     94 ;                  Make BUS return line BFS parameter if system manager
0000     95 ;                  wants to override the executor buffer size on a per-line
0000     96 ;                  basis.
0000     97 ;                  Conditionally allow the DMC to perform I/O functions on the
0000     98 ;                  controller.
0000     99 ;
0000    100 ;          V012    TMH0012         Tim Halvorsen   20-Dec-1982
0000    101 ;                  Add new UNA parameter to table of parameters to ignore
0000    102 ;                  on sensemode requests.
0000    103 ;                  If the error qualifier returned by a datalink driver is
0000    104 ;                  "physical address", meaning that DECnet's requested physical
0000    105 ;                  address is in conflict with another user of the device,
0000    106 ;                  then map the qualifier to "hardware address", for which
0000    107 ;                  a NICE parameter exists.  This is so that when an NCP
0000    108 ;                  user requests to have the line turned on, and it fails
0000    109 ;                  due to address conflict, then a parameter is returned with
0000    110 ;                  "bad parameter value".
0000    111 ;                  Suppress node addresses if areas have been hidden.
0000    112 ;
0000    113 ;          V011    TMH0011         Tim Halvorsen   29-Nov-1982
0000    114 ;                  Remove one lagging reference to NMA$C_CIRTY_LAP symbol.
```

```
0000   115 ;                Add support for Ethernet protocol type parameter, rather
0000   116 ;                than using a hard-coded constant.  This is so that a UNA
0000   117 ;                can be used on another protocol type without interfering
0000   118 ;                with the normal DECnet traffic on an experimental basis.
0000   119 ;
0000   120 ;        V010    TMH0010         Tim Halvorsen   13-Oct-1982
0000   121 ;                Journal I/O requests to PSIACP.
0000   122 ;                Replace all occurrences of NMA$C_CIRTY_LAP with
0000   123 ;                NMA$C_CIRTY_LAPB.
0000   124 ;                Remove defaulting of USAGE to OUTGOING if NUMBER
0000   125 ;                is set, since NUMBER now is used for both incoming
0000   126 ;                and outgoing DLM circuits.
0000   127 ;                Fix bug in SET LINE if mnemonic doesn't parse correctly.
0000   128 ;
0000   129 ;        V009    TMH0009         Tim Halvorsen   27-Sep-1982
0000   130 ;                Change meaning of LPD DRT cell from the designated
0000   131 ;                router's node address to it's ADJ index.
0000   132 ;                Modify circuit substate action routine so that value
0000   133 ;                is returned if circuit is in "S" state.
0000   134 ;
0000   135 ;        V008    TMH0008         Tim Halvorsen   14-Sep-1982
0000   136 ;                Deassign channel if error detected trying to start
0000   137 ;                a broadcast line and the executor type is not Phase IV.
0000   138 ;                (channel was getting left assigned).
0000   139 ;                If an INCOMING DLM circuit is defined, then declare
0000   140 ;                NETACP as able to receive incoming calls, in case it
0000   141 ;                wasn't already defined in the SET EXECUTOR (PSI might
0000   142 ;                not have been up then).
0000   143 ;                Make USAGE default to OUTGOING if the NUMBER parameter
0000   144 ;                is set on a DLM circuit.
0000   145 ;                Add DLM circuit defaulting (separate from transport defaults).
0000   146 ;                Add support for automatic line counters and automatic PSI
0000   147 ;                native circuit counters.
0000   148 ;
0000   149 ;        V007    TMH0007         Tim Halvorsen   31-Aug-1982
0000   150 ;                Add action routine to return designated router on NI.
0000   151 ;                Fix counter returns for DLM switched circuits.
0000   152 ;
0000   153 ;        V006    TMH0006         Tim Halvorsen   01-Jul-1982
0000   154 ;                Add Phase IV line, circuit support.
0000   155 ;                Add journalling of SETMODE QIOWs.
0000   156 ;                Fix error path in PSI counter sensing.
0000   157 ;                Modify PNA, PNN, BLO and add LIT to work with adjacencies.
0000   158 ;                Add PLI DEVNAM action routine, which returns the physical
0000   159 ;                device name being used by the line, in VMS format, including
0000   160 ;                the unit number.
0000   161 ;                Do not allow user to turn on an NI line device unless
0000   162 ;                the executor type is set to Phase IV, since Phase III
0000   163 ;                doesn't handle broadcast circuits.
0000   164 ;                Fix bug in preallocation of receive buffers - it wasn't
0000   165 ;                including the CXB overhead in the buffer size.
0000   166 ;                Clear PLI MCD before each SHOW function.
0000   167 ;
0000   168 ;        V005    TMH0005         Tim Halvorsen   30-Jun-1982
0000   169 ;                Change defaulting of line and circuit parameters, to
0000   170 ;                be dependant on the type of line or circuit.  This is
0000   171 ;                done because the default parameters for a DDCMP circuit
```

```
0000   172 ;                are not the same as for a X25 circuit, or an NI circuit.
0000   173 ;                The same thing is true for DDCMP or LAPB lines.
0000   174 ;                Add code to ask PSI for it's line/circuit parameters on
0000   175 ;                SHOW functions, so that a show returns the sum for both
0000   176 ;                NETACPs and PSIs databases.
0000   177 ;                Change mneumonic of KMS to KMX, marking it "multiple-unit
0000   178 ;                controller" and add KMY.
0000   179 ;
0000   180 ;        V004    TMH0004         Tim Halvorsen   16-Jun-1982
0000   181 ;                Always give null context area on PSI control QIOs.
0000   182 ;                Fix list of line/circuit parameters which are passed to PSI.
0000   183 ;                Add support for asking PSI for its line/circuit counters.
0000   184 ;                Fix bug in sensing of datalink counters, to correctly
0000   185 ;                detect and remove the "seconds since last zeroed" counter
0000   186 ;                (it never worked, but no driver ever returned it either).
0000   187 ;                Don't tell PSI about SVCs, since it doesn't want to know
0000   188 ;                about them.
0000   189 ;                Add kludge to SEND_TO_PSI, so that all ACPCONTROL functions
0000   190 ;                to PSIACP are terminated with NFB$C_ENDOFLIST, to avoid the
0000   191 ;                check which doesn't allow 0 field identifiers.
0000   192 ;
00^0   193 ;        V003    TMH0003         Tim Halvorsen   15-Jun-1982
0000   194 ;                Remove code to disable error reporting from PSI control QIOs.
0000   195 ;
0000   196 ;        V002    TMH0002         Tim Halvorsen   04-Apr-1982
0000   197 ;                Make this new module from the CRI and PLI routines in CNFACT
0000   198 ;                in an attempt to reduce the size of that module.
0000   199 ;                Remove code to make the start key canonical, since
0000   200 ;                the start key no longer exists.
0000   201 ;                Add code to check if we are dealing with a LAPB line,
0000   202 ;                and if so, give it to PSIACP.
0000   203 ;                Remove some obsolete symbols.
0000   204 ;                Remove X.25 parameters from NICE-NFB translation table,
0000   205 ;                since we will never have to translate PSI NICE parameters.
0000   206 ;                Clean up code in INSERT_CRI and INSERT_PLI.
0000   207 ;                Fix code to "update" volatile fields in CNF from values given
0000   208 ;                by the datalink driver, so that it doesn't stop when it can't
0000   209 ;                modify a particular field, but continues thru all the rest of
0000   210 ;                the parameters returned by the driver.  A single field which
0000   211 ;                is conditionally writable might stop the code from ever
0000   212 ;                processing the rest of the fields.
0000   213 ;                Make default word displacement.
0000   214 ;                Send DELETE requests to PSI.
0000   215 ;                Clean up SHOW_xxx routines.
0000   216 ;                Pre-allocate receive buffers before issuing a SETMODE
0000   217 ;                startup function to a datalink driver, so that pool
0000   218 ;                expansion can be done to accomodate the buffers.  It
0000   219 ;                is done here because pool expansion can't be done on
0000   220 ;                the interrupt stack in the datalink driver.
0000   221 ;                Default the line protocol based on the line name mnemonic.
0000   222 ;                Add X25 to the list of allowed mnemonics for circuit names.
0000   223 ;                Send all circuits set/clears with TYPE=X25 to PSI.
0000   224 ;                Change all CNF action routines to use the new action routine
0000   225 ;                interface (NETCNF now automatically allocates a TMP buffer).
0000   226 ;                Remove obsolete NUL action routines.
0000   227 ;                Do not send "show-only" datalink parameters (returned by the
0000   228 ;                driver and stored into the CNF) back to the driver in the
```

```
0000   229 ;                next setmode.  Mark show-only datalink parameters in the
0000   230 ;                translation table.
0000   231 ;                Make circuits of the form X25-x default to TYPE=X25 so that
0000   232 ;                you don't have to enter TYPE all the time.
0000   233 ;                Reformat secondary search key into cononical form if the
0000   234 ;                field is the line or circuit name.
0000   235 ;                Add KMS to device list for X.25 network management.
0000   236 ;                Add X.25 datalink mapping support.
0000   237 ;                Allow any characters to appear after circuit names of the
0000   238 ;                form "X25-xxx"
0000   239 ;
0000   240 ;        V001    TMH0001         Tim Halvorsen   27-Mar-1982
0000   241 ;                Fix code to translate an NMA parameter code returned by
0000   242 ;                a datalink driver validation error to a NFB code.
0000   243 ;---
```

```
                     0000   245          .SBTTL  DECLARATIONS
                     0000   246 ;
                     0000   247 ; INCLUDE FILES:
                     0000   248 ;
                     0000   249          $DEVDEF         ; Define device characteristics bits
                     0000   250          $DVIDEF         ; $GETDVI item codes
                     0000   251          $DCDEF          ; Device classs definitions
                     0000   252          $CCBDEF         ; Channel Control Block
                     0000   253          $CXBDEF         ; Complex-chained buffers
                     0000   254          $UCBDEF         ; Unit Control Block
                     0000   255          $XMDEF          ; XMDRIVER (general communications device driver)
                     0000   256
                     0000   257          $NFBDEF         ; Network Function Block (ACP control QIO definitions)
                     0000   258          $NMADEF         ; Network Management (NICE protocol) definitions
                     0000   259          $EVCDEF         ; DECnet Event logging symbols
                     0000   260          $CNRDEF         ; Configuration Root block
                     0000   261          $CNFDEF         ; Configuration data block
                     0000   262          $DEVTRNDEF      ; Device translation (VMS vs. Nice protocol) info
                     0000   263          $DLLQIODEF      ; Data-Link Layer QIO definitions
                     0000   264          $RCBDEF         ; Routing Control Block (analogous to Volume Control
                     0000   265                          ; block)
                     0000   266          $LPDDEF         ; Logical path descriptor
                     0000   267          $ADJDEF         ; Adjacency control block
                     0000   268          $NETSYMDEF      ; Miscellaneous network symbols
                     0000   269          $NETUPDDEF      ; Symbols used in private NETACP interface to NETDRIVER
                     0000   270          $NSPMSGDEF      ; DNA architecture definitions & message formats
                     0000   271
                     0000   272 ;
                     0000   273 ; Macros
                     0000   274 ;
                     0000   275
                     0000   276          .MACRO  SWAP,ARG1,ARG2          ; Swap arguments in-place
                     0000   277          XORL    ARG1,ARG2
                     0000   278          XORL    ARG2,ARG1
                     0000   279          XORL    ARG1,ARG2
                     0000   280          .ENDM
                     0000   281
                     0000   282 ;
                     0000   283 ; Define the following symbol to allow controller I/O functions on the DMC.
                     0000   284 ;
                     0000   285 ;DMC_MODEM      = 0
                     0000   286
                     0000   287 ;
                     0000   288 ; Define special CNF flags for each database
                     0000   289 ;
                     0000   290
00000004             0000   291 PLI_V_NEW       = CNF$V_FLG_MRK1        ; New PLI being built for first time
```

```
                     0000   293 ;
                     0000   294 ; OWN STORAGE
                     0000   295 ;
                     0000   296
                  00000000  297         .PSECT  NET_PURE,NOWRT,NOEXE,LONG
                     0000   298
                     0000   299 ;
                     0000   300 ;  The following macroes build a conversion table for formatting counters
                     0000   301 ;  into NICE format.  Each counter i.d. contain is bit encoded to contain
                     0000   302 ;  formatting information as follows:
                     0000   303 ;
                     0000   304 ;   15   14    13  12   11          0          Bit
                     0000   305 ;
                     0000   306 ;  < 1 >< width >< 0 >< counter i.d. >         Field
                     0000   307 ;
                     0000   308 ;
               00000001  0000   309         $WIDTH_B = 1    ; Counter width specifier for bytes
               00000002  0000   310         $WIDTH_W = 2    ; Counter width specifier for words
               00000003  0000   311         $WIDTH_L = 3    ; Counter width specifier for longwords
                     0000   312
               0000C000  0000   313 NET$C_NMACNT_SLZ = <1@15>!<<$WIDTH_W>@13>!0     ; Seconds since last zeroed
                     0000   314
                     0000   315 .MACRO $COUNT_ENT  base,nice,pre,mod,count,width; Insert table entry
                     0000   316                                                  ;
                     0000   317         .WORD   <1@15>!<<$WIDTH_'width'>@13>!-   ; Counter flag, Counter width
                     0000   318                 <NMA$C_'nice'_'count'>           ; Nice counter i.d.
                     0000   319         .WORD   'pre'$'width'_'mod''count' -     ; Offset into internal structure
                     0000   320                 - base                          ; minus internal structure base
                     0000   321 .ENDM   $COUNT_ENT                               ;
                     0000   322
                     0000   323
                     0000   324 .MACRO  $COUNT_TAB  base,nice,pre,mod,list       ; Create counter formatting table
                     0000   325                                                  ;
                     0000   326         .IRP    A,<list>                        ;
                     0000   327                 $COUNT_ENT  base,nice,pre,mod,A ; Insert table entry
                     0000   328         .ENDR                                   ;
                     0000   329                                                 ;
                     0000   330         .LONG   0                               ; Terminate the table
                     0000   331
                     0000   332 .ENDM   $COUNT_TAB
                     0000   333
                     0000   334
                     0000   335 LPD_CNT_TAB:                     ; Circuit counter formatting table
                     0000   336
                     0000   337     $COUNT_TAB  LPD$L_ABS_TIM,CTCIR,LPD,CNT_,-
                     0000   338                 <-
                     0000   339                 <APR,L>,-       ; Arriving packets received
                     0000   340                 <DPS,L>,-       ; Departing packets sent
                     0000   341                 <ACL,W>,-       ; Arriving congestion loss
                     0000   342                 <TPR,L>,-       ; Transit packets received
                     0000   343                 <TPS,L>,-       ; Transit packets sent
                     0000   344                 <TCL,W>,-       ; Transit congestion loss
                     0000   345                 <LDN,B>,-       ; Line down events
                     0000   346                 <IFL,B>,-       ; Initialization failures
                     0000   347     >
                     0024   348
                     0024   349 X25_CNT_TAB:                     ; X.25 native circuit counter formatting table
```

```
                        ;
0024    351     $COUNT_TAB  ,,,,            ; Currently only "seconds since last zeroed"
0028    352                                 ; is maintained outside PSIACP
0028    353
0028    354 PLI_CNT_TAB:                    ; Line counter formatting table
0028    355
0028    356     $COUNT_TAB  ,,,,            ; Currently only "seconds since last zeroed"
002C    357                                 ; is maintained outside the driver
002C    358
```

```
                     002C    360 ;
                     002C    361 ;  Build a table to driver the translation between Network Management (NICE)
                     002C    362 ;  device names to VMS devices names and to supply miscellaneous information
                     002C    363 ;  about known devices
                     002C    364 ;
FFFFFFFF             002C    365 NMA$C_LINPR_ = -1                           ; Define protocol type for protocols
                     002C    366                                             ; not defined by the NICE protocol
                     002C    367
0000000F             002C    368 MAX_C_DEVNAM = 15                           ; Maximum device name string
                     002C    369                                             ; (dev-c-u.t)
                     002C    370
                     002C    371 .MACRO  .devtrn netman,vms,type,prot,char       ; Create table entry
                     002C    372
                     002C    373         $$devtrn = .                        ; Remember where we are
                     002C    374         .ascic  "NETMAN"                    ; Enter Net Man device mnemonic
                     002C    375         . = $$devtrn + devtrn$b_vms         ; Advance to next field
                     002C    376         .ascic  "VMS"                       ; Enter VMS device mnemonic
                     002C    377         . = $$devtrn + devtrn$b_dev         ; Advance to next field
                     002C    378         .byte   devtrn$c_dev_'TYPE'
                             379         . = $$devtrn + devtrn$b_prot        ; Advance to next field
                     002C    380         .byte   nma$c_linpr_'PROT'          ; Enter default protocol type
                     002C    381         . = $$devtrn + devtrn$b_char        ; Advance to device characteristics
                     002C    382         .if nb,char                         ; Enter device characteristics
                     002C    383                 .byte   devtrn$m_'CHAR'
                     002C    384         .iff
                     002C    385                 .byte   0
                     002C    386         .endc
                     002C    387         . = $$devtrn + devtrn$c_length      ; Advance to next table entry
                     002C    388
                     002C    389 .ENDM   .devtrn
                     002C    390
                     002C    391 DEVTRN_TABLE:                               ; Device translation table
                     002C    392
                     002C    393         .devtrn <DMC>, <XM>, <DMC>, <POI>, <>           ; DMC-11
                     0039    394         .devtrn <DMP>, <XD>, <DMP>, <POI>, <>           ; DMP-11
                     0046    395         .devtrn <PCL>, <XP>, <PCL>, <MAS>, <>           ; PCL-11
                     0053    396         .devtrn <DMF>, <XG>, <DMF>, <POI>, <>           ; DMF-11 (combo board)
                     0060    397         .devtrn <CI>,  <CN>, <CI>,  <>,    <>           ; CI-780
                     006D    398         .devtrn <UNA>, <XE>, <UNA>, <NI>,  <>           ; UNA (Ethernet)
                     007A    399         .devtrn <PPUNA>,<_XE>,<PPUNA>,<POI>, <MULTI>    ; "point-to-point" UNA
                     0087    400                                                         ; (internal only)
                     0087    401         ; "_XE" must be used to distinguish PPUNA datalinks from UNA datalinks,
                     0087    402         ; since they both use the same driver, and since matching of lines and
                     0087    403         ; and circuits is done by the VMS device name given here.
                     0087    404         .devtrn <QNA>, <XQ>, <UNA>, <NI>,  <>           ; QNA (Ethernet)
                     0094    405         .devtrn <UE>,  <XX>, <UNA>, <NI>,  <>           ; 3COM UE (Ethernet)
                     00A1    406         .devtrn <DUP>, <>,   <DUP>, <LAPB>, <>          ; DUP-11 (for X.25)
                     00AE    407         .devtrn <DPV>, <>,   <DUP>, <LAPB>, <>          ; DPV-11 (for X.25)
                     00BB    408         .devtrn <KMX>, <>,   <KMS>, <LAPB>, <MULTI>     ; KMS-11 (for X.25)
                     00C8    409         .devtrn <KMY>, <>,   <KMS>, <LAPB>, <>          ; KMS-11 (for X.25)
                     00D5    410         .devtrn <KMV>, <>,   <KMS>, <LAPB>, <>          ; KMV-11 (for X.25)
                     00E2    411         .devtrn <X25>, <NW>, <X25>, <>,    <>           ; X25 DLM circuits
                     00EF    412         .devtrn <>,    <>,   <UNK>, <>,    <MULTI>      ; Terminate the table
                     00FC    413                                                         ; with foreign device
                     00FC    414         .ALIGN  LONG
```

```
                     00FC   416 ;
                     00FC   417 ;  The following tables contain a list of parameters which need to be cleared
                     00FC   418 ;  in a given (PLI or CRI) CNF block before that block is process for a "show"
                     00FC   419 ;  QIO.  The reason for this is that the parameters are maintained only by the
                     00FC   420 ;  device driver and the copy of this information stored in the CNF is stale.
                     00FC   421 ;
                     00FC   422 PLI_CLR_TAB:    .CNFFLD pli,s,mcd       ; X.25 microcode dump file [write only]
00000000             0100   423                 .LONG   0               ; Terminate the table
                     0104   424
                     0104   425 CRI_CLR_TAB:    .CNFFLD cri,l,pls       ; Polling sub-state
00000000             0108   426                 .LONG   0               ; Terminate the table
                     010C   427
                     010C   428 ;
                     010C   429 ; The following table lists those CRI parameters which must be processed by
                     010C   430 ; PSIACP if the circuit is of TYPE X25.  This table is used to build
                     010C   431 ; a control QIO to send the parameters to PSIACP.  PSIACP only needs to "know"
                     010C   432 ; about the parameters listed in this table.
                     010C   433 ;
                     010C   434
                     010C   435 CRI_PSI_TAB:
                     010C   436         .CNFFLD cri,l,STA       ; State
                     0110   437         .CNFFLD cri,l,USE       ; Circuit usage
                     0114   438         .CNFFLD cri,l,TYP       ; Circuit type
                     0118   439         .CNFFLD cri,l,CHN       ; Channel
                     011C   440         .CNFFLD cri,l,MBL       ; Maximum data
                     0120   441         .CNFFLD cri,l,MWI       ; Maximum window
                     0124   442         .CNFFLD cri,s,DTE       ; DTE
                     0128   443
00000000             0128   444         .LONG   0               ; Terminate table
00000020             012C   445 CRI_PSI_SIZ = . - CRI_PSI_TAB
                     012C   446
                     012C   447 ;
                     012C   448 ; The following table lists those PLI parameters which must be processed by
                     012C   449 ; PSIACP if the line is using PROTOCOL LAPB.  This table is used to build
                     012C   450 ; a control QIO to send the parameters to PSIACP.  PSIACP only needs to "know"
                     012C   451 ; about the parameters listed in this table.
                     012C   452 ;
                     012C   453
                     012C   454 PSI_PLI_CLR_TAB:                  ; Params cleared before every SET function
                     012C   455         .CNFFLD pli,l,sub       ; Line substate
00000000             0130   456         .LONG   0               ; Terminate the table
                     0134   457
                     0134   458 PLI_PSI_TAB:
                     0134   459         .CNFFLD pli,v,DUP       ; Duplex
                     0138   460         .CNFFLD pli,v,CON       ; Controller mode
                     013C   461         .CNFFLD pli,l,STA       ; State
                     0140   462         .CNFFLD pli,l,SUB       ; Substate [SHOW only]
                     0144   463         .CNFFLD pli,l,PRO       ; Protocol
                     0148   464         .CNFFLD pli,l,HTI       ; Holdback timer
                     014C   465         .CNFFLD pli,l,MBL       ; Maximum block
                     0150   466         .CNFFLD pli,l,MRT       ; Maximum retransmits
                     0154   467         .CNFFLD pli,l,MWI       ; Maximum window
                     0158   468         .CNFFLD pli,l,BFN       ; Receive buffers
                     015C   469         .CNFFLD pli,l,RTT       ; Retransmit timer
                     0160   470         .CNFFLD pli,l,MOD       ; X.25 mode (DCE, DTE, etc.)
                     0164   471         .CNFFLD pli,s,MCD       ; Microcode dump filespec  [SET only]
                     0168   472
```

```
00000000  0168   473          .LONG   0                       ; Terminate table
00000038  016C   474 PLI_PSI_SIZ = . - PLI_PSI_TAB
          016C   475
          016C   476 ;
          016C   477 ;  Build the tables which drive datalink QIO parameter translation.  The
          016C   478 ;  datalink QIO parameter buffers are stored in NICE protocol format.
          016C   479 ;
00003FFF  016C   480 CNVTAB$C_INTRNL = ^X3FFF                    ; Index = INTRNL means datalink only
          016C   481
          016C   482 $VIELD  CNVTAB,0,<-                        ; Define conversion table enter
          016C   483                   <NMA,12,M>,-             ; NICE protocol param i.d.
          016C   484                   <RO,1,M>,-               ; Read-only flag
          016C   485                   <,3>,-                   ; 3 spare bits
          016C   486                   <INT,14,M>,-             ; Internal parameter index
          016C   487                   <FMT, 2,M>,-             ; Parameter format (string,value,etc)
          016C   488          >
          016C   489
          016C   490 .MACRO  .cnvtab nma,db,param,readonly=0 ; Insert parameter i.d. table entry
          016C   491
          016C   492          .LONG   <<nma$c_'nma'_'param'>            @ <cnvtab$v_nma>> -
          016C   493                + <'readonly'                       @ cnvtab$v_ro> -
          016C   494                + <<nfb$c_'db'_'param' & nfb$m_inx> @ <cnvtab$v_int-nfb$v_inx>> -
          016C   495                + <<nfb$c_'db'_'param' & nfb$m_typ> @ <cnvtab$v_fmt-nfb$v_typ>>
          016C   496 .ENDM
          016C   497
          016C   498 .MACRO  .inttab nma,typ,param            ; Describe internal NMA codes
          016C   499                                          ; returned by datalink drivers
          016C   500          .LONG   <nma$c_'nma'_'param'              @ cnvtab$v_nma> -
          016C   501                + <cnvtab$c_intrnl                  @ cnvtab$v_int> -
          016C   502                + <nfb$c_typ_'typ'                  @ cnvtab$v_fmt>
          016C   503 .ENDM
          016C   504
          016C   505 CRI_TRN_TAB:                             ; NICE-internal param i.d. translation
          016C   506          .cnvtab pcci,cri,POL            ; Polling state
          0170   507          .cnvtab pcci,cri,PLS,1          ; Polling sub-state  [READ ONLY]
          0174   508          .cnvtab pcci,cri,TYP,1          ; Protocol type  [READ ONLY]
          0178   509          .cnvtab pcci,cri,TRI            ; Tributary station address
          017C   510          .cnvtab pcci,cri,BBT            ; Babbel timer
          0180   511          .cnvtab pcci,cri,TRT            ; Transmit timer
          0184   512          .cnvtab pcci,cri,MRB            ; Maximum receive buffers
          0188   513          .cnvtab pcci,cri,MTR            ; Maximum transmits
          018C   514          .cnvtab pcci,cri,ACB            ; Active base
          0190   515          .cnvtab pcci,cri,ACI            ; Active increment
          0194   516          .cnvtab pcci,cri,IAB            ; Inactive base
          0198   517          .cnvtab pcci,cri,IAI            ; Inactive increment
          019C   518          .cnvtab pcci,cri,IAT            ; Inactive threshold
          01A0   519          .cnvtab pcci,cri,DYB            ; Dying base
          01A4   520          .cnvtab pcci,cri,DYI            ; Dying increment
          01A8   521          .cnvtab pcci,cri,DYT            ; Dying threshold
          01AC   522          .cnvtab pcci,cri,DTH            ; Dead treshold
          01B0   523          .cnvtab pcci,cri,MST            ; Maintenance mode state
          01B4   524
00000000  01B4   525          .LONG   0                       ; Terminate the table
          01B8   526
          01B8   527
          01B8   528 PLI_TRN_TAB:                             ; NICE-internal param i.d. translation
          01B8   529          .cnvtab pcli,pli,BUS            ; Receive buffer size
```

```
                01BC    530         .cnvtab pcli,pli,DUP    ; Duplex mode
                01C0    531         .cnvtab pcli,pli,CON    ; Contrc'ler (loopback) mode
                01C4    532         .crvtab pcli,pli,CLO    ; Clock mode
                01C8    533         .cnvtab pcli,pli,PRO    ; Protocol
                01CC    534         .cn tab pcli,pli,SLT    ; Scheduling timer
                01D0    535         .cnvtab pcli,pli,DDT    ; Dead timer
                01D4    536         .cnvtab pcli,pli,DLT    ; Delay timer
                01D8    537         .cnvtab pcli,pli,SRT    ; Stream timer
                01DC    538         .cnvtab pcli,pli,BFN    ; Number of buffers in pool
                01E0    539         .cnvtab pcli,pli,RTT    ; Retransmit timer
                01E4    540         .cnvtab pcli,pli,HWA,1  ; UNA hardware address  [READ ONLY]
                01E8    541
00000000        01E8    542         .LONG   0               ; Terminate the table
                01EC    543
```

```
                    00000000   545          .PSECT  NET_IMPURE,WRT,NOEXE
                          0000 546
                    00000001  0000 547 PLI_B_STATE:    .BLKB   1               ; Device state
00000000 00000000   00000000  0001 548 PLI_Q_DEVNAM:   .QUAD   0               ; Device name descriptor
                          0009 549
                    00000020  0009 550 DEVNAM_C_SIZ = 32                       ; Size of SEARCH key buffer
                          0009 551                                             ; for line and circuit names
                          0009 552
                    00000029  0009 553 SRCH_BUF:       .BLKB   DEVNAM_C_SIZ    ; Optional SEARCH KEY buffer
                          0029 554
00000000 00000000   00000000  0029 555 QIOW_Q_IOSB:    .QUAD   0               ; General QIOW IOSB
                    00000033  0031 556 QIOW_W_RETLEN:  .BLKW   1               ; Return length word for PSI QIOW
00000000 00000000   00000000  0033 557 TMP_Q_DESC:     .QUAD   0               ; Scratch descrirptor
                          0038 558
                    00000000   559          .PSECT  NET_CODE,NOWRT,EXE
```

```
                        561          .SBTTL  NET$SCAN_xxx - Scan database
                        562 ;+
                        563 ; NET$SCAN_CRI - Scan CRI database
                        564 ; NET$SCAN_PLI - Scan PLI database
                        565 ;
                        566 ; This co-routine is used to scan the database, and return to the caller
                        567 ; (co-routine) for each entry in the database.  These routines establish
                        568 ; the order of the database entries, above that of the natural ordering of
                        569 ; the collating field.
                        570 ;
                        571 ; Inputs:
                        572 ;
                        573 ;       R11 = Address of CNR
                        574 ;       R10 = Address of starting CNF (or 0 if to start at the beginning)
                        575 ;
                        576 ; Outputs:
                        577 ;
                        578 ;       R10 = Address of CNF if dialogue aborted prematurely, else 0.
                        579 ;
                        580 ; The caller receives control on each database entry in list (via co-routine
                        581 ; call).
                        582 ;
                        583 ; On input to co-routine:
                        584 ;
                        585 ;       R0 = True if entry was found.  False if at end of list (R10 invalid)
                        586 ;       R10 = Address of CNF entry found
                        587 ;
                        588 ; On output from co-routine:
                        589 ;
                        590 ;       R0 = CNF$_ADVANCE       Advance to next CNF, continue dialogue
                        591 ;            CNF$_TAKE_PREV     Return previous CNF, abort dialogue
                        592 ;            CNF$_TAKE_CURR     Return current CNF, abort dialgoue
                        593 ;            CNF$_QUIT          Return no CNF (R10 = 0), abort dialogue
                        594 ;
                        595 ;  ***  These routines must be abortable via a RET  ***
                        596 ;---
                        597
                        598 NET$SCAN_CRI::
                        599 NET$SCAN_PLI::
FFFD'  31               600          BRW     DEFAULT_SCAN    ; Use default scanner with collating field
```

```
                     0003   602          .SBTTL  NET$PRE_QIO_xxx - Pre-QIO processing
                     0003   603 ;+
                     0003   604 ; NET$PRE_QIO_CRI - Perform pre-QIO CRI database processing
                     0003   605 ; NET$PRE_QIO_PLI - Perform pre-QIO PLI database processing
                     0003   606 ;
                     0003   607 ; This routine is called just after validating the NFB for a database
                     0003   608 ; function to do any special pre-processing before the request is attempted.
                     0003   609 ;
                     0003   610 ; Here, for CRI and PLI databases, we reformat the search key into a standard
                     0003   611 ; format for line and circuit names, so that we don't get two database entries
                     0003   612 ; for DMC-0 and DMC-0.0, which should refer to the same entry.
                     0003   613 ;
                     0003   614 ; Inputs:
                     0003   615 ;
                     0003   616 ;       R11 = Address of CNR
                     0003   617 ;       NET$GQ_SRCH_KEY = Descriptor of search key value
                     0003   618 ;       NET$GL_SRCH_ID = Field ID of search field.
                     0003   619 ;       NET$GQ_SRCH2_KEY = Descriptor of secondary search key value
                     0003   620 ;       NET$GL_SRCH2_ID = Field ID of secondary search field.
                     0003   621 ;
                     0003   622 ; Outputs:
                     0003   623 ;
                     0003   624 ;       NET$GQ_SRCH_KEY = Reformatted search key value
                     0003   625 ;       NET$GQ_SRCH2_KEY = Reformatted secondary search key value
                     0003   626 ;---
                     0003   627
                     0003   628 NET$PRE_QIO_CRI::                        ;
                     0003   629         $CNFFLD cri,s,nam,R9             ; Get circuit name field i.d.
          07   11    000A   630         BRB     PREQIO_PRI_CRI           ; Continue in common
                     000C   631
                     000C   632 NET$PRE_QIO_PLI::                        ;
                     000C   633         $CNFFLD pli,s,nam,R9             ; Get line name field i.d.
                     0013   634                                          ; Fall thru to common code
                     0013   635 PREQIO_PRI_CRI:
                     0013   636         ;
                     0013   637         ;   Rebuild SEARCH key if needed
                     0013   638         ;
00000000'EF    59    D1     0013   639         CMPL    R9,NET$GL_SRCH_ID        ; Is the SEARCH key the name?
               29    12     001A   640         BNEQ    10$                      ; If NEQ no
57    00000000'EF    7D     001C   641         MOVQ    NET$GQ_SRCH_KEY,R7       ; Get the SEARCH KEY descriptor
53    00000009'EF    9E     0023   642         MOVAB   SRCH_BUF,R3              ; Point to the SEARCH KEY buffer
00000000'EF    53    CE     002A   643         MNEGL   R3,NET$GQ_SRCH_KEY       ; Prepare for size calculation
00000004'EF    53    D0     0031   644         MOVL    R3,NET$GQ_SRCH_KEY+4     ; Point to new SEARCH KEY string
             0040    30     0038   645         BSBW    REBUILD_NAME             ; Rebuild the circuit name
           3C 50     E9     003B   646         BLBC    R0,210$                  ; If LBC then error in name
00000000'EF    53    C0     003E   647         ADDL    R3,NET$GQ_SRCH_KEY       ; Calculate size of the string
                     0045   648         ;
                     0045   649         ;   Rebuild secondary SEARCH key if needed
                     0045   650         ;
00000000'EF    59    D1     0045   651 10$:     CMPL    R9,NET$GL_SRCH2_ID       ; Is the secondary SEARCH key the name?
               29    12     004C   652         BNEQ    100$                     ; If NEQ no
57    00000000'EF    7D     004E   653         MOVQ    NET$GQ_SRCH2_KEY,R7      ; Get the secondary search key descriptor
53    00000009'EF    9E     0055   654         MOVAB   SRCH_BUF,R3              ; Point to the SEARCH KEY buffer
00000000'EF    53    CE     005C   655         MNEGL   R3,NET$GQ_SRCH2_KEY      ; Prepare for size calculation
00000004'EF    53    D0     0063   656         MOVL    R3,NET$GQ_SRCH2_KEY+4    ; Point to new SEARCH KEY string
             000E    30     006A   657         BSBW    REBUILD_NAME             ; Rebuild the circuit name
           0A 50     E9     006D   658         BLBC    R0,210$                  ; If LBC then error in name
```

```
00000000'EF    53   C0  0070    659          ADDL    R3,NET$GQ_SRCH2_KEY     ; Calculate size of the string
          50    00'  D0  0077    660 100$:    MOVL    S^#SS$_NORMAL,R0        ; Indicate success
                     05  007A    661 210$:    RSB                             ; Return status in R0
                         007B    662
                         007B    663
                         007B    664 REBUILD_NAME:                           ; Rebuild the name to canonical form
                         007B    665          ;
                         007B    666          ;   Rebuild the device name so that it is of the form:
                         007B    667          ;
                         007B    668          ;       'dev-c-u.t'  or  'dev-c-u'     for multiplexed device types
                         007B    669          ;
                         007B    670          ;       'dev-c.t'    or  'dev-c'       for non-multiplexed device types
                         007B    671          ;
                         007B    672          ;
                         007B    673          ;
          20    57   D1  007B    674          CMPL    R7,#DEVNAM_C_SIZ        ; Output name will be LEQ input name,
                60   1A  007E    675          BGTRU   40$                     ; If GTRU unsigned then may oveflow buf
                         0080    676          ;
                         0080    677          ;   Move the device mnemonic and locate the device table entry
                         0080    678          ;
            133A   30    0080    679          BSBW    PRS_MNEMONIC            ; Parse name, locate the table entry
          5A 50    E9    0083    680          BLBC    R0,40$                  ; If LBC the error
    63    64 55    28    0086    681          MOVC3   R5,(R4),(R3)            ; Enter the Net. Man. device mnemonic
                         008A    682          ;
                         008A    683          ;   If the name is of the form "X25-xxx", then allow any remaining
                         008A    684          ;   characters to be accepted as is.  The user must make sure that
                         008A    685          ;   the name is spelled exactly the same for all requests.
                         008A    686          ;
      08   0A A6    91    008A    687          CMPB    DEVTRN$B_DEV(R6),#DEVTRN$C_DEV_X25 ; X.25 circuit name?
                0A   12    008E    688          BNEQ    1$                      ; Branch if not
           83   2D   90    0090    689          MOVB    #^A"-",(R3)+            ; Move delimiter
    63     68   57   28    0093    690          MOVC    R7,(R8),(R3)            ; Copy in rest of name
              0073   31    0097    691          BRW     90$                     ; Exit successfully
                         009A    692 1$:      ;
                         009A    693          ;   Move the controller i.d. -- suppress leading zeroes.  If the device
                         009A    694          ;   is not a "MULTI-plexed" device then the units field may not be
                         009A    695          ;   present and the name may be end with the controller specifier field
                         009A    696          ;   or it may end with either the units field or the tributary field or
                         009A    697          ;   both.  If the device is "MULTI-plexed" then the name must contain
                         009A    698          ;   at least the units field following the controller specifier field.
                         009A    699          ;
           54   2D   9A    009A    700          MOVZBL  #^A"-",R4               ; Setup the field delimiter
           83   54   90    009D    701          MOVB    R4,(R3)+                ; Enter it into the output buffer
                00   E0    00A0    702          BBS     #DEVTRN$V_MULTI,-       ; If BS then multi-unit device
                14 0C A6    00A2    703                    DEVTRN$B_CHAR(R6),10$ ;
           50   57   7D    00A5    704          MOVQ    R7,R0                   ; Copy string descriptor
                50   D7    00A8    705 5$:      DECL    R0                      ; Any characters left
                0D   19    00AA    706          BLSS    10$                     ; If LSS then no
           61   2D   91    00AC    707          CMPB    #^A"-",(R1)             ; Delimited by a "-" ?
                08   13    00AF    708          BEQL    10$                     ; If EQL yes
           81   2E   91    00B1    709          CMPB    #^A".",(R1)+            ; Delimited by a "." ?
                F2   12    00B4    710          BNEQ    5$                      ; If NEQ then keep trying
           54   2E   9A    00B6    711          MOVZBL  #^A".",R4               ; Reset delimiter for parse
              134A   30    00B9    712 10$:     BSBW    PRS_DECIMAL             ; Get binary value of controller
             21 50   E9    00BC    713          BLBC    R0,40$                  ; If LBC then error
                52   D5    00BF    714          TSTL    R2                      ; Any decimal characters parsed?
                1D   13    00C1    715          BEQL    40$                     ; If EQL no, error in name
```

```
                50    51   D0  00C3   716          MOVL    R1,R0                   ; Setup binary value to be converted
                   FF37'   30  00C6   717          BSBW    NET$BIN2ASC             ; Convert to ascii and insert
                           00C9   718          ;
                           00C9   719          ;   Move the unit number  -- suppress leading zeros.
                           00C9   720          ;
                     00    E0  00C9   721          BBS     #DEVTRN$V_MULTI,-       ; If BS then multi-unit device
                14 0C  A6        00CB   722                   DEVTRN$B_CHAR(R6),50$ ;
                           00CE   723          ;
                           00CE   724          ;   Single unit device -- verify unit number is zero but don't move it
                           00CE   725          ;
                54    2E   91  00CE   726          CMPB    #^A".",R4               ; Did we already rule out a unit field?
                     25    13  00D1   727          BEQL    70$                     ; If EQL then yes
                54    2E   9A  00D3   728          MOVZBL  #^A".",R4               ; Setup the field delimiter
                   132D    30  00D6   729          BSBW    PRS_DECIMAL             ; Get binary unit number
                  36 50    E9  00D9   730          BLBC    R0,100$                 ; If LBC then error
                     51    D5  00DC   731          TSTL    R1                      ; Single unit device, is it unit 0?
                     18    13  00DE   732          BEQL    70$                     ; If EQL yes, process tributary field
                     30    11  00E0   733  40$:    BRB     100$                    ; Else error in name
                           00E2   734  50$:
                           00E2   735          ;   Device is a multi-unit device, move unit number field
                           00E2   736          ;
                83    2D   90  00E2   737          MOVB    #^A"-",(R3)+            ; Enter prefix to unit number field
                54    2E   9A  00E5   738          MOVZBL  #^A".",R4               ; Setup the field delimiter
                   131B    30  00E8   739          BSBW    PRS_DECIMAL             ; Get binary unit number
                  24 50    E9  00EB   740          BLBC    R0,100$                 ; If EQL then error
                     52    D5  00EE   741          TSTL    R2                      ; Any decimal characters parsed?
                     20    13  00F0   742          BEQL    100$                    ; If EQL then error
                50    51   D0  00F2   743          MOVL    R1,R0                   ; Get binary value for ascii conversion
                   FF08'   30  00F5   744          BSBW    NET$BIN2ASC             ; Move as ascii @R3
                           00F8   745  70$:
                           00F8   746          ;   Process the tributary field.  If this is the CRI database then move
                           00F8   747          ;   the remainder of the name (unfortunately we cannot determine if a
                           00F8   748          ;   tributary specifier is valid for this device yet since we cannot
                           00F8   749          ;   determine for sure if the device is to be run in the multi-drop or
                           00F8   750          ;   point-to-point mode until later on in the QIO cycle).
                           00F8   751          ;
                     04    91  00F8   752          CMPB    #NFB$C_DB_CRI,-         ; Circuit database ?
                  0A AB            00FA   753                  CNR$B_TYPE(R11)  ;
                     04    13  00FC   754          BEQL    80$                     ; If EQL yes
                     57    D5  00FE   755          TSTL    R7                      ; Else there must be no characters left
                     10    12  0100   756          BNEQ    100$                    ; If NEQ then illegal name
                     57    D5  0102   757  80$:    TSTL    R7                      ; Any characters left?
                     07    13  0104   758          BEQL    90$                     ; If EQL no, return success
                83    2E   90  0106   759          MOVB    #^A".",(R3)+            ; Enter delimiter
             63 68    57   28  0109   760          MOVC3   R7,(R8),(R3)            ; Enter remaining characters, update R3
                50    00'  D0  010D   761  90$:    MOVL    S^#SS$_NORMAL,R0        ; Indicate success
                     05    11  0110   762          BRB     110$                    ; Take common exit
                           0112   763
             50 0000'8F    3C  0112   764  100$:   MOVZWL  #SS$_IVDEVNAM,R0        ; Indicate invalid device name
                           05  0117   765  110$:   RSB                             ; Return status in R0
```

```
                     0118   767          .SBTTL  NET$SHOW_xxx - Pre-SHOW processing
                     0118   768 ;+
                     0118   769 ; NET$SHOW_CRI - Show QIO pre-processing for CRI database
                     0118   770 ; NET$SHOW_PLI - Show QIO pre-processing for PLI database
                     0118   771 ;
                     0118   772 ; This routine is called for each CNF which is about to be returned
                     0118   773 ; to a "show" QIO.  For the CRI and PLI databases, we must "update"
                     0118   774 ; the fields reflecting datalink driver status, by issuing a SENSEMODE
                     0118   775 ; QIO to the driver, and taking it's current values for datalink parameters
                     0118   776 ; and storing them into the CNF.  This makes both the CNF and the driver
                     0118   777 ; consistent.
                     0118   778 ;
                     0118   779 ; Inputs:
                     0118   780 ;
                     0118   781 ;       R11 = Address of CNR
                     0118   782 ;       R10 = Address of CNF
                     0118   783 ;
                     0118   784 ; Outputs:
                     0118   785 ;
                     0118   786 ;       R0 = Status code
                     0118   787 ;
                     0118   788 ;       The CNF may be updated.
                     0118   789 ;-
                     0118   790
                     0118   791 NET$SHOW_CRI::                          ; "Show" QIO pre-processing for one CNF
     55 00000104'EF   9E  0118   792          MOVAB   CRI_CLR_TAB,R5      ; Get table of params to clear
                023E   30  011F   793          BSBW    CLEAR_VOLATILE      ; Clear volatile parameters
                           0122   794          ;
                           0122   795          ;   If this is an X.25 circuit, then ask PSI for its parameter values
                           0122   796          ;
                           0122   797          $GETFLD cri,l,typ           ; Get type of circuit
              08 50    E9  012F   798          BLBC    R0,5$               ; Branch if not set
           03    58    D1  0132   799          CMPL    R8,#NMA$C_CIRTY_X25 ; X.25 circuit?
              03    12  0135   800          BNEQ    5$                  ; Branch if not
             0147   31  0137   801          BRW     GET_PSI_CRI         ; Get current PSI parameter values
                           013A   802          ;
                           013A   803          ;   Issue SENSEMODE to driver, and store results in CRI
                           013A   804          ;
              FEC3'  30  013A   805 5$:      BSBW    NET$LOCATE_LPD      ; Get LPD associated with this CRI
              18 50    E9  013D   806          BLBC    R0,10$              ; If EQL then none
        58    28 A6    9A  0140   807          MOVZBL  LPD$B_PLVEC(R6),R8  ; Get PLVEC index
              12       13  0144   808          BEQL    10$                 ; Skip if none assigned
        53    14 A6    3C  0146   809          MOVZWL  LPD$W_CHAN(R6),R3   ; Get I/O channel
     54    0000'8F    3C  014A   810          MOVZWL  #IO$_SENSEMODE,R4   ; Get I/O function code
56   0000016C'EF    9E  014F   811          MOVAB   CRI_TRN_TAB,R6      ; Get table for NICE param translation
              46       11  0156   812          BRB     SHO_PLI_CRI         ; Pre-process the CRI CNF block
                           0158   813
           50    01    D0  0158   814 10$:      MOVL    #1,R0               ; Report success - return parameters
                       05  015B   815          RSB
                           015C   816
                           015C   817 NET$SHOW_PLI::                          ; "Show" QIO pre-processing for one CNF
55   000000FC'EF    9E  015C   818          MOVAB   PLI_CLR_TAB,R5      ; Get table of params to clear
              01FA   30  0163   819          BSBW    CLEAR_VOLATILE      ; Clear volatile parameters
                           0166   820          ;
                           0166   821          ; If this is a LAPB line, ask PSIACP for it's parameter values
                           0166   822          ;
                           0166   823          $GETFLD pli,l,pro           ; Get protocol parameter value
```

```
                08 50    E9  0173   824          BLBC    R0,5$                           ; If not specified, skip it
             05    58    D1  0176   825          CMPL    R8,#NMA$C_LINPR_LAPB            ; LAPB line?
                03       12  0179   826          BNEQ    5$                              ; Branch if not
                0149     31  017B   827          BRW     GET_PSI_PLI                     ; Get current PSI PLI parameter values
                         017E   828          ;
                         017E   829          ;   Ask the datalink driver for it's parameter values
                         017E   830          ;
          50    12 AA    3C  017E   831 5$:     MOVZWL  CNF$W_ID(R10),R0                ; Get PLVEC index
                16       13  0182   832          BEQL    10$                             ; If EQL then none assigned
    53  00000000'EF40    B0  0184   833          MOVW    PLVEC$AW_CHAN[R0],R3            ; Get I/O channel
      54    0000'8F      B0  018C   834          MOVW    #IO$_SENSEMODE!IO$M_CTRL,R4 ; Get QIO function code
    56    000001B8'EF    9E  0191   835          MOVAB   PLI_TRN_TAB,R6                  ; Get table for NICE param translation
                04       11  0198   836          BRB     SHO_PLI_CRI
                         019A   837
          50    01       D0  019A   838 10$:    MOVL    #1,R0                           ; Report success - return parameters
                05       RSB 019D   839
                         019E   840
                         019E   841 SHO_PLI_CRI:
                         019E   842          ;
                         019E   843          ;   Get a list of parameters from the driver
                         019E   844          ;
       00000029'EF       7C  019E   845          CLRQ    QIOW_Q_IOSB                     ; Init IOSB image
                         01A4   846          $QIOW_S -                               ; Issue QIO to get device info
                         01A4   847                  EFN  = #NET$C_EFN_WAIT,-
                         01A4   848                  CHAN = R3,-
                         01A4   849                  FUNC = R4,-
                         01A4   850                  IOSB = QIOW_Q_IOSB,-
                         01A4   851                  P2   = #TMPBUF_DESC
             1A 50       E9  01C7   852          BLBC    R0,100$                         ; If LBC as ume unsupported by driver
    13 00000029'EF       E9  01CA   853          BLBC    QIOW_Q_IOSB,100$                ; If LBC assume unsupported by driver
0B 0000002D'EF   0B       E1  01D1   854          BBC     #XM$V_STS_ACTIVE,QIOW_Q_IOSB+4,100$ ; Skip if datalink not active
   54    00000004'EF       D0  01D9   855          MOVL    TMPBUF_DESC+4,R4                ; Point to parameter list
                06       10  01E0   856          BSBB    JAM_CNF                         ; Stuff the params into the CNF block
                03       11  01E2   857          BRB     110$                            ; Take common exit
          50    01       D0  01E4   858 100$:   MOVL    #1,R0                           ; Report success
                05       RSB 01E7   859 110$:   RSB                                     ; Return status in R0
```

```
                                01E8    861          .SBTTL  JAM_CNF - Store driver values into CNF
                                01E8    862 ;+
                                01E8    863 ; JAM_CNF - Store datalink driver parameters into CNF
                                01E8    864 ;
                                01E8    865 ; This routine updates the CNF with the latest and greatest parameter
                                01E8    866 ; values from the datalink driver.
                                01E8    867 ;
                                01E8    868 ; Inputs:
                                01E8    869 ;
                                01E8    870 ;       R11 = Address of CNR
                                01E8    871 ;       R10 = Address of CNF
                                01E8    872 ;       R6 = Address of datalink-NICE translation table
                                01E8    873 ;       R4 = Address of buffer returned by SENSEMODE
                                01E8    874 ;       QIOW_Q_IOSB = IOSB from SENSEMODE operation
                                01E8    875 ;
                                01E8    876 ; Outputs:
                                01E8    877 ;
                                01E8    878 ;       R0 = Status code
                                01E8    879 ;-
                                01E8    880          .ENABL  LSB
                                01E8    881
                                01E8    882 JAM_CNF:
                00000000'EF   DD 01E8    883          PUSHL   NET$GL_FLAGS            ; Save current flags
                                01EE    884          CLRBIT  NET$V_CNFLCK,NET$GL_FLAGS ; Allow update of cond. write fields
                                01F6    885          ;
                                01F6    886          ;   Jam the values from the QIO buffer one at time into the CNF block.
                                01F6    887          ;
                0000002B'EF   B5 01F6    888 10$:     TSTW    QIOW_Q_IOSB+2           ; Any params left?
                          66   13 01FC    889          BEQL    110$                    ; If EQL we're done
                          70   10 01FE    890          BSBB    TAKE_2_IOSB             ; See if we have two bytes left
                      59  84   3C 0200    891          MOVZWL  (R4)+,R9                ; Get NICE protocol param i.d. + type
                      55  56   D0 0203    892          MOVL    R6,R5                   ; Point to translation table
                        1136   30 0206    893          BSBW    CVT_NMA_INT             ; Convert NMA i.d. to internal i.d.
                     2C 50   E9 0209    894          BLBC    R0,70$                  ; If LBC then not found
02      59      02    10   ED 020C    895          CMPZV   #NFB$V_TYP,#NFB$S_TYP,R9,#NFB$C_TYP_STR ; Is it a string ?
                          0A   13 0211    896          BEQL    50$                     ; If EQL yes
                               0213    897          ;
                               0213    898          ;   It's a longword
                               0213    899          ;
                      57  04   D0 0213    900          MOVL    #4,R7                   ; We'll take 4 more bytes
                          5B   10 0216    901          BSBB    TAKE_R7_IOSB            ; Deplete IOSB count
                      58  84   D0 0218    902          MOVL    (R4)+,R8                ; Get the value
                          0F   11 021B    903          BRB     60$                     ; Continue
                               021D    904          ;
                               021D    905          ;   It's a string
                               021D    906          ;
                          51   10 021D    907 50$:     BSBB    TAKE_2_IOSB             ; Account for count field
                      57  84   32 021F    908          CVTWL   (R4)+,R7                ; Get string size
                          45   19 0222    909          BLSS    120$                    ; If LSS then string is too long
                      58  54   D0 0224    910          MOVL    R4,R8                   ; Point to string
                      54  57   C0 0227    911          ADDL    R7,R4                   ; Skip over it
                          47   10 022A    912          BSBB    TAKE_R7_IOSB            ; Deplete bytes for string
              3FFF    8F  59   B1 022C    913 60$:     CMPW    R9,#CNVTAB$C_INTRNL     ; Datalink only parameter?
                          C3   13 0231    914          BEQL    10$                     ; If so, just ignore it
                    FDCA'  30 0233    915          BSBW    CNF$PUT_FIELD           ; Store it in the CNF
                               0236    916                                          ; Ignore errors - field might be R/O
                          BE   11 0236    917          BRB     10$                     ; Loop through all parameters
```

```
                        0238   918          ;
                        0238   919          ;  Parameter is not found i  tables, skip it if present in buffer
                        0238   920          ;
0000002B'EF   B5        0238   921 70$:     TSTW    QIOW_Q_IOSB+2           ; Deplete bytes
 0B 59    0C    E0      023E   922          BBS     #12,R9,80$              ; Br if string value
                        0242   923          ;
                        0242   924          ;       It's a longword
                        0242   925          ;
     57  04    D0      0242   926          MOVL    #4,R7                   ; We'll take 4 more bytes
           2C  10      0245   927          BSBB    TAKE_R7_IOSB            ; Deplete IOSB count
     54  04    C0      0247   928          ADDL    #4,R4                   ; Skip valued parameter
         FFA9  31      024A   929          BRW     10$                     ; Get rest of buffer
                        024D   930          ;
                        024D   931          ;       It's a string
                        024D   932          ;
           21  10      024D   933 80$:     BSBB    TAKE_2_IOSB             ; Account for count field
     57    84  32      024F   934          CVTWL   (R4)+,R7                ; Get string size
           15  19      0252   935          BLSS    120$                    ; If LSS then string is too long
     54    57  C0      0254   936          ADDL    R7,R4                   ; Skip over it
           1A  10      0257   937          BSBB    TAKE_R7_IOSB            ; Deplete bytes for string
         FF9A  31      0259   938          BRW     10$                     ; Get rest of buffer
                        025C   939
00000000'EF   8ED0      025C   940 100$:    POPL    NET$GL_FLAGS            ; Restore flags
                05      0263   941          RSB                             ; Return status in R0
                        0264   942
     50  01    D0      0264   943 110$:    MOVL    #1,R0                   ; Return success
           F3  11      0267   944          BRB     100$
                        0269   945
   50  0000'8F   3C      0269   946 120$:    MOVZWL  #SS$_RESULTOVF,R0       ; Indicate buffer error
           EC  11      026E   947          BRB     100$
                        0270   948
                        0270   949 TAKE_2_IOSB:
     57  02    3C      0270   950          MOVZWL  #2,R7                   . Setup bytes to take
                        0273   951 TAKE_R7_IOSB:
0000002B'EF  57    A2  0273   952          SUBW    R7,QIOW_Q_IOSB+2        ; Deplete bytes
           04  18      027A   953          BGEQ    200$                    ; If GEQ then okay
           8E  D5      027C   954          TSTL    (SP)+                   ; Pop caller's return address
           E9  11      027E   955          BRB     120$                    ; Exit with buffer error
                05      0280   956 200$:    RSB
                        0281   957
                        0281   958          .DSABL  LSB
```

```
                     0281   960          .SBTTL  GET_PSI_xxx - Get current PSI parameter values
                     0281   961 ;+
                     0281   962 ; GET_PSI_CRI - Get current PSI circuit parameter values
                     0281   963 ; GET_PSI_PLI - Get current PSI line parameter values
                     0281   964 ;
                     0281   965 ; This routine is called to get the current PSI parameter values
                     0281   966 ; from it's database, and store them in ours, so the SHOW function returns
                     0281   967 ; the sum of both databases.
                     0281   968 ;
                     0281   969 ; Inputs:
                     0281   970 ;
                     0281   971 ;       R11 = CRI CNR address
                     0281   972 ;       R10 = CRI CNF address
                     0281   973 ;
                     0281   974 ; Outputs:
                     0281   975 ;
                     0281   976 ;       R0 = status
                     0281   977 ;-
                     0281   978 GET_PSI_CRI:
                     0281   979          ;
                     0281   980          ;   If this is a PVC, then go ahead and ask PSI.  If it's a
                     0281   981          ;   incoming or outgoing SVC, then don't bother, since PSI
                     0281   982          ;   doesn't know anything about them.
                     0281   983          ;
                     0281   984          $GETFLD cri,l,use                ; Get USAGE parameter
  05 50    E9        028E   985          BLBC    R0,5$                    ; Branch if not set - defaulting error
00   58    D1        0291   986          CMPL    R8,#NMA$C_CIRUS_PER      ; PVC?
     04    13        0294   987          BEQL    10$                      ; If not, then exit successfully
50   01    D0        0296   988 5$:      MOVL    #1,R0                    ; Do nothing
           05        0299   989          RSB
                     029A   990          ;
                     029A   991          ;   Issue a SHOW function to PSI for its circuit parameters
                     029A   992          ;
                     029A   993 10$:     $CNFFLD cri,s,nam,R9             ; Set field ID of circuit name
           0697      30     02A1   994          BSBW    SEND_TO_PSI              ; Call co-routine to setup NFB, etc.
         1F 50       E9     02A4   995          BLBC    R0,90$                   ; Branch if error detected
      66   22        90     02A7   996          MOVB    #NFB$C_FC_SHOW,NFB$B_FCT(R6) ; Set function code
           3C        BB     02AA   997          PUSHR   #^M<R2,R3,R4,R5>         ; Save registers
                     02AC   998          ASSUME  NFB$C_ENDOFLIST EQ 0     ; NFB terminator and PSI_TAB are same
62 0000010C'EF      20     28 02AC   999          MOVC    #CRI_PSI_SIZ,CRI_PSI_TAB,(R2)  ; Copy in the field IDs
           3C        BA     02B4  1000          POPR    #^M<R2,R3,R4,R5>         ; Restore registers
        52   20 A2    9E     02B6  1001          MOVAB   CRI_PSI_SIZ(R2),R2       ; Increment field ID list pointer
   53   00000200'8F    C0     02BA  1002          ADDL    #P4BUF_SIZE,R3           ; Set size of P4 return buffer
           9E        16     02C1  1003          JSB     @(SP)+                   ; Call SEND_TO_PSI back to issue QIO
         2E 50       E8     02C3  1004          BLBS    R0,STORE_PSI_PARAMS      ; Branch if ok
           05        02C6  1005 90$:     RSB                              ; Return with error
                     02C7  1006
                     02C7  1007 GET_PSI_PLI:
                     02C7  1008          ;
                     02C7  1009          ;   Issue a SHOW function to PSI for its line parameters
                     02C7  1010          ;
                     02C7  1011          $CNFFLD pli,s,nam,R9             ; Set field ID of circuit name
           066A      30     02CE  1012          BSBW    SEND_TO_PSI              ; Call co-routine to setup NFB, etc.
         1F 50       E9     02D1  1013          BLBC    R0,90$                   ; Branch if error detected
      66   22        90     02D4  1014          MOVB    #NFB$C_FC_SHOW,NFB$B_FCT(R6) ; Set function code
           3C        BB     02D7  1015          PUSHR   #^M<R2,R3,R4,R5>         ; Save registers
                     02D9  1016          ASSUME  NFB$C_ENDOFLIST EQ 0     ; NFB terminator and PSI_TAB are same
```

```
        62  00000134'EF    38   28  02D9  1017          MOVC    #PLI_PSI_SIZ,PLI_PSI_TAB,(R2)  ; Copy in the field IDs
                           3C   BA  02E1  1018          POPR    #^M<R2,R3,R4,R5>               ; Restore registers
                     52    38 A2  9E  02E3  1019          MOVAB   PLI_PSI_SIZ(R2),R2             ; Increment field ID list pointer
                53   00000200'8F  C0  02E7  1020          ADDL    #P4BUF_SIZE,R3                 ; Set size of P4 return buffer
                           9E   16  02EE  1021          JSB     @(SP)+                         ; Call SEND_TO_PSI back to issue QIO
                        01 50   E8  02F0  1022          BLBS    R0,STORE_PSI_PARAMS            ; Branch if ok
                                05  02F3  1023 90$:      RSB                                    ; Return with error
                                    02F4  1024
                                    02F4  1025 STORE_PSI_PARAMS:
                                    02F4  1026          ;
                                    02F4  1027          ;   Copy the parameters returned by PSI into our database
                                    02F4  1028          ;
                     54    57   7D  02F4  1029          MOVQ    R7,R4                          ; Make copy of result descriptor
                  56    10 A6  9E  02F7  1030          MOVAB   NFB$L_FLDID(R6),R6             ; Point to field ID list
                     0200 8F  BB  02FB  1031          PUSHR   #^M<R9>                        ; Save registers
                00000000'EF  DD  02FF  1032          PUSHL   NET$GL_FLAGS                   ; Save ACP flags
                                0305  1033          CLRBIT  NET$V_CNFLCK,NET$GL_FLAGS ; Allow update of cond. write fields
                     59    86   D0  030D  1034 20$:      MOVL    (R6)+,R9                       ; Get next field ID
                           34   13  0310  1035          BEQL    80$                            ; Branch if end of list
                02   59    02   10  ED  0312  1036          CMPZV   #NFB$V_TYP,#NFB$S_TYP,R9,#NFB$C_TYP_STR ; Is it a string?
                           0C   13  0317  1037          BEQL    30$                            ; Branch if so
                     54    C4   C2  0319  1038          SUBL    #4,R4                          ; Decrement buffer space left
                           21   19  031C  1039          BLSS    70$                            ; Branch if not enough space returned
                     58    85   D0  031E  1040          MOVL    (R5)+,R8                       ; Get longword value
                           EA   19  0321  1041          BLSS    20$                            ; Branch if "cleared"
                           15   11  0323  1042          BRB     50$                            ; Store parameter
                     54    02   C2  0325  1043 30$:      SUBL    #2,R4                          ; Decrement space left
                           15   19  0328  1044          BLSS    70$                            ; Branch if not enough space returned
                     57    85   3C  032A  1045          MOVZWL  (R5)+,R7                       ; Get length of string
                           DE   13  032D  1046          BEQL    20$                            ; Branch if "cleared"
                     54    57   C2  032F  1047          SUBL    R7,R4                          ; Decrement space left
                           0B   19  0332  1048          BLSS    70$                            ; Branch if not enough space returned
                     58    55   D0  0334  1049          MOVL    R5,R8                          ; Point to start of string
                     55    57   C0  0337  1050          ADDL    R7,R5                          ; Skip past string
                        FCC3'   30  033A  1051 50$:      BSBW    CNF$PUT_FIELD                  ; Store the parameter
                           CE   11  033D  1052          BRB     20$                            ; Loop thru all parameters
                                    033F  1053
        50    0000'8F    3C   033F  1054 70$:      MOVZWL  #SS$_RESULTOVF,R0              ; Set overflow status
                           03   11  0344  1055          BRB     90$
                                0346  1056
                     50    01   D0  0346  1057 80$:      MOVL    #1,R0
                     51    50   D0  0349  1058 90$:      MOVL    R0,R1                          ; Save final status
         00000000'EF  8ED0  034C  1059          POPL    NET$GL_FLAGS                   ; Restore ACP flags
                     50  8ED0  0353  1060          POPL    R0                             ; Point to scratch storage
         00000000'EF    16   0356  1061          JSB     NET$DEALLOCATE                 ; Deallocate SHOW buffer
                     50    51   D0  035C  1062          MOVL    R1,R0                          ; Restore final status
                                05  035F  1063          RSB
```

```
                     0360  1065          .SBTTL  CLEAR_VOLATILE - Clea  list of volatile parameters
                     0360  1066 ;+
                     0360  1067 ; CLEAR_VOLATILE - Clear list of volatile parameters from a CNF
                     0360  1068 ;
                     0360  1069 ; This routine is called with a list of parameters to be cleared so
                     0360  1070 ; that stale data is not kept in a CNF block.
                     0360  1071 ;
                     0360  1072 ; Inputs:
                     0360  1073 ;
                     0360  1074 ;       R11 = Address of CNR
                     0360  1075 ;       R10 = Address of CNF
                     0360  1076 ;       R5 = Address of list of longword parameter IDs
                     0360  1077 ;
                     0360  1078 ; Outputs:
                     0360  1079 ;
                     0360  1080 ;       None
                     0360  1081 ;-
                     0360  1082 CLEAR_VOLATILE:
59   85   D0         0360  1083 10$:    MOVL    (R5)+,R9                ; Get next parameter i.d.
     05   13         0363  1084         BEQL    20$                     ; If EQL then none left
 FC98'    30         0365  1085         BSBW    CNF$CLR_FIELD           ; Clear param i.. CNF, ignore errors
     F6   11         0368  1086         BRB     10$                     ; Loop
          05         036A  1087 20$:    RSB
```

```
                        0368  1089          .SBTTL  NET$DEFAULT_CRI - Apply default values
                        0368  1090 ;+
                        0368  1091 ; NET$DEFAULT_CRI  -  Apply default values to selected CNF parameters.
                        0368  1092 ;
                        0368  1093 ; This routine is called by CNF$INSERT just prior to validating a CNF
                        0368  1094 ; entry which is to be inserted into the database.  Its purpose is to
                        0368  1095 ; supply default values to selected parameters.
                        0368  1096 ;
                        0368  1097 ; INPUTS:        R11     CNR pointer
                        0368  1098 ;                R10     CNF pointer
                        0368  1099 ;
                        0368  1100 ; OUTPUTS:       R11     CNR pointer
                        0368  1101 ;                R10     CNF pointer
                        0368  1102 ;                R0      Status code.
                        0368  1103 ;
                        0368  1104 ;                All other registers contain garbage.
                        0368  1105 ;-
                        0368  1106 NET$DEFAULT_CRI::
          FC92'  30     0368  1107          BSBW    NET$APPLY_DFLT          ; Apply standard table defaults
                        036E  1108          $GETFLD cri,s,nam               ; Get descriptor of circuit name
           103F  30     037B  1109          BSBW    PRS_MNEMONIC            ; Locate the table entry
         75 50  E9      037E  1110          BLBC    R0,TO$                  ; If not found, assume DDCMP
                        0381  1111          ;
                        0381  1112          ;   If the circuit type is not specified yet, and the name is
                        0381  1113          ;   the form X25-x, then apply the default X25 circuit parameters
                        0381  1114          ;   (including setting it's type to "X25").
                        0381  1115          ;
      08   0A A6   91   0381  1116          CMPB    DEVTRN$B_DEV(R6),#DEVTRN$C_DEV_X25 ; Is circuit X25-x?
              6F   12   0385  1117          BNEQ    10$                     ; If not, skip it
   56 00000000'EF   9E   0387  1118          MOVAB   NET$G_CRI_X25,R6        ; Get address of X25 circuit defaults
            FC6F'  30   038E  1119          BSBW    NET$TABLE_DFLT          ; Apply the default values
                        0391  1120          ;
                        0391  1121          ;   If the USAGE is INCOMING or OUTGOING, then assume that
                        0391  1122          ;   this circuit is to be used for X.25 datalink mapping, since
                        0391  1123          ;   native users of X.25 circuits don't ever set incoming or
                        0391  1124          ;   outgoing circuits.
                        0395  1125          ;
                        0396  1126          $GETFLD cri,l,use               ; Get USAGE parameter
          1A 50   E9    039A  1127          BLBC    R0,5$                   ; If not set, skip it
                        03A1  1128          $DISPATCH R8,<-                 ; Based on Usage,
                        03A1  1129                  <NMA$C_CIRUS_INC,4$>-   ; If incoming
                        03A1  1130                  <NMA$C_CIRUS_OUT,4$>>   ; or outgoing, then set DLM on
             10   11   03A9  1131          BRB     5$                      ; Else, skip it
          58   01   D0   03AB  1132 4$:     MOVL    #1,R8                   ; Set "true"
                        03AE  1133          $PUTFLD cri,v,dlm               ; Enable X.25 datalink mapping
                        03BB  1134          ;
                        03BB  1135          ;   If X.25 datalink mapping is enabled, then apply the Transport
                        03BB  1136          ;   defaults.
                        03BB  1137          ;
                        03BB  1138 5$:      $GETFLD cri,v,dlm               ; Is X.25 datalink mapping enabled?
          51 50   E9    03C8  1139          BLBC    R0,80$                  ; Branch if not
          4E 58   E9    03CB  1140          BLBC    R8,80$
   56 00000000'EF   9E   03CE  1141          MOVAB   NET$G_CRI_DLM,R6        ; Get address of DLM defaults
            FC28'  30   03D5  1142          BSBW    NET$TABLE_DFLT          ; Apply the default values
                        03D8  1143          $GETFLD cri,l,use               ; Get circuit usage
          02   58   D1   03E5  1144          CMPL    R8,#NMA$C_CIRUS_OUT     ; Outgoing circuit?
               28   12   03E8  1145          BNEQ    20$                     ; Skip if not
```

```
                56  00000000'EF   9E  03EA  1146          MOVAB   NET$G_CRI_DLMOUT,R6     ; Get address of outgoing DLM defaults
                          FC0C'   30  03F1  1147          BSBW    NET$TABLE_DFLT          ; Apply the default values
                             1C   11  03F4  1148          BRB     20$                     ; then apply Transport defaults
                                  03F6  1149          ;
                                  03F6  1150          ;   If it's a NI circuit, then apply the appropriate defaults
                                  03F6  1151          ;
                 09   0A A6       91  03F6  1152  10$:   CMPB    DEVTRN$B_DEV(R6),#DEVTRN$C_DEV_UNA ; Is it an NI circuit?
                                  12  03FA  1153          BNEQ    15$                     ; If not, skip it
                56  00000000'EF   9E  03FC  1154          MOVAB   NET$G_CRI_NI,R6         ; Get address of NI circuit defaults
                          FBFA'   30  0403  1155          BSBW    NET$TABLE_DFLT          ; Apply the default values
                             0A   11  0406  1156          BRB     20$                     ; Apply Transport defaults as well
                                  0408  1157          ;
                                  0408  1158          ; For DDCMP circuits, apply another set of default values
                                  0408  1159          ;
                56  00000000'EF   9E  0408  1160  15$:   MOVAB   NET$G_CRI_DDCMP,R6       ; Get address of DDCMP circuit defaults
                          FBEE'   30  040F  1161          BSBW    NET$TABLE_DFLT          ; Apply the default values
                                  0412  1162          ;
                                  0412  1163          ;   For circuits used by Transport, then apply the Transport defaults.
                                  0412  1164          ;
                56  00000000'EF   9E  0412  1165  20$:   MOVAB   NET$G_CRI_TRN,R6         ; Get address of Transport defaults
                          FBE4'   30  0419  1166          BSBW    NET$TABLE_DFLT          ; Apply the default values
                          50   01   D0  041C  1167  80$:   MOVL    #1,R0                   ; Return successful
                                  05  041F  1168          RSB
```

```
                                                H  1
                     0420  1170          .SBTTL  NET$DEFAULT_PLI - Apply default values
                     0420  1171 ;+
                     0420  1172 ; NET$DEFAULT_PLI  -  Apply default values to selected CNF parameters.
                     0420  1173 ;
                     0420  1174 ; This routine is called by CNF$INSERT just prior to validating a CNF
                     0420  1175 ; entry which is to be inserted into the database.  Its purpose is to
                     0420  1176 ; supply default values to selected parameters.
                     0420  1177 ;
                     0420  1178 ; INPUTS:        R11     CNR pointer
                     0420  1179 ;                 R10     CNF pointer
                     0420  1180 ;
                     0420  1181 ; OUTPUTS:       R11     CNR pointer
                     0420  1182 ;                 R10     CNF pointer
                     0420  1183 ;                 R0      Status code
                     0420  1184 ;
                     0420  1185 ;                 All other registers contain garbage.
                     0420  1186 ;-
                     0420  1187 NET$DEFAULT_PLI::
           FBDD'  30 0420  1188          BSBW    NET$APPLY_DFLT          ; Apply standard table defaults
                     0423  1189          $GETFLD pli,s,nam               ; Get descriptor of line name
            0F8A  30 0430  1190          BSBW    PRS_MNEMONIC            ; Locate the table entry
         01 50  E8 0433  1191          BLBS    R0,10$                  ; If found, proceed
                05 0436  1192          RSB                             ; Else, not found
                     0437  1193 10$:     ;
                     0437  1194          ;   If the protocol is not specified yet, stuff it's default value
                     0437  1195          ;   based on the line name nmemonic.
                     0437  1196          ;
                     0437  1197          $GETFLD pli,l,pro               ; Was protocol explicitly specified?
              0D 50  E8 0444  1198          BLBS    R0,20$                  ; If so, don't change it
           58 0B A6  9A 0447  1199          MOVZBL  DEVTRN$B_PROT(R6),R8    ; Get default protocol type
         FF 8F   58  91 044B  1200          CMPB    R8,#NMA$C_LINPR_        ; Is there a default protocol?
                03  13 044F  1201          BEQL    20$                     ; Branch if not
              FBAC'  30 0451  1202          BSBW    CNF$PUT_FIELD           ; Store the protocol into the CNF
                     0454  1203          ;
                     0454  1204          ;   Based on the type of line protocol, apply a different set of
                     0454  1205          ;   defaults for each type.
                     0454  1206          ;
              0A A6  91 0454  1207 20$:    CMPB    DEVTRN$B_DEV(R6),-      ; Is this a "point-to-point" XE?
                 0A           0457  1208                  #DEVTRN$C_DEV_PPUNA
                 3C  13 0458  1209          BEQL    50$                     ; If so, use NI defaults
                     045A  1210          $GETFLD pli,l,pro               ; Get line protocol type
              4E 50  E9 0467  1211          BLBC    R0,80$                  ; If none, then no defaults
                     046A  1212          $DISPATCH R8,<-                 ; Based on line protocol,
                     046A  1213                  <NMA$C_LINPR_POI,30$>-  ; DDCMP Point-to-point
                     046A  1214                  <NMA$C_LINPR_CON,30$>-  ; DDCMP Controller station
                     046A  1215                  <NMA$C_LINPR_TRI,30$>-  ; DDCMP Tributary
                     046A  1216                  <NMA$C_LINPR_DMC,30$>-  ; DDCMP DMC compatible mode
                     046A  1217                  <NMA$C_LINPR_LAPB,40$>- ; X.25 LAPB protocol
                     046A  1218                  <NMA$C_LINPR_NI,50$>>   ; NI protocol
                 3A  11 047C  1219          BRB     80$                     ; If none of the above, then skip it
                     047E  1220          ;
                     047E  1221          ;   Apply DDCMP defaults
                     047E  1222          ;
   56  00000000'EF   9E 047E  1223 30$:    MOVAB   NET$G_PLI_DDCMP,R6      ; Get address of DDCMP defaults
              FB78'  30 0485  1224          BSBW    NET$TABLE_DFLT          ; Apply default values
                 2E  11 0488  1225          BRB     80$
                     048A  1226          ;
```

```
                                048A  1227          ;   Apply LAPB defaults
                                048A  1228          ;
          56  00000000'EF   9E    048A  1229 40$:     MOVAB   NET$G_PLI_LAPB,R6       ; Get address of LAPB defaults
                    FB6C'   30    0491  1230          BSBW    NET$TABLE_DFLT          ; Apply default values
                      22    11    0494  1231          BRB     80$
                                0496  1232          ;
                                0496  1233          ;   Apply NI defaults
                                0496  1234          ;
          56  00000000'EF   9E    0496  1235 50$:     MOVAB   NET$G_PLI_NI,R6         ; Get address of NI defaults
                    FB60'   30    049D  1236          BSBW    NET$TABLE_DFLT          ; Apply default tables
                                04A0  1237          ;
                                04A0  1238          ;   For NI lines, add the "hardware address" parameter at this time,
                                04A0  1239          ;   so that there is enough space reserved in the CNF block for future
                                04A0  1240          ;   replacements of the value by the datalink layer.  Initially, it
                                04A0  1241          ;   will have the value of 6 bytes of binary zero.  This is done here
                                04A0  1242          ;   because the defaulting tables don't handle string defaults.
                                04A0  1243          ;
                          7E    7C    04A0  1244          CLRQ    -(SP)                   ; Zero 8 byte buffer
                    57    06    D0    04A2  1245          MOVL    #6,R7                   ; Set length of string to 6 bytes
                    58    5E    D0    04A5  1246          MOVL    SP,R8                   ; Point to buffer of 8 null bytes
                                04A8  1247          $PUTFLD pli,s,hwa               ; Allocate space for NI hardware address
                    5E    08    C0    04B5  1248          ADDL    #8,SP                   ; Deallocate buffer
                    50    01    D0    04B8  1249 80$:     MOVL    #1,R0                   ; Return successful
                                05    04BB  1250 90$:     RSB
```

```
                                04BC  1252          .SBTTL  NET$INSERT_CRI - Pre-insertion processing
                                04BC  1253 ;+
                                04BC  1254 ; NET$INSERT_CRI - Insert CRI entry into database
                                04BC  1255 ;
                                04BC  1256 ; This routine is called to validate a CRI CNF entry before inserting
                                04BC  1257 ; it into the database.
                                04BC  1258 ;
                                04BC  1259 ; Inputs:
                                04BC  1260 ;
                                04BC  1261 ;       R11 = Address of CNR
                                04BC  1262 ;       R10 = Address of CNF
                                04BC  1263 ;
                                04BC  1264 ; Outputs:
                                04BC  1265 ;
                                04BC  1266 ;       R0 = Status code.  If error, entry is not inserted.
                                04BC  1267 ;-
                                04BC  1268 NET$INSERT_CRI::
                      7E    D4  04BC  1269          CLRL    -(SP)                   ; Scratch space on stack for STATE
                50    0000'8F   3C  04BE  1270          MOVZWL  #SS$_INSFARG,R0         ; Set error if parameter not present
                                04C3  1271          $GETFLD cri,l,sta               ; Get the operator state
                      03 50     E8  04D0  1272          BLBS    R0,5$                   ; Error if not already defaulted
                        009A    31  04D3  1273          BRW     200$                    ; Exit with status
                      6E    58  D0  04D6  1274 5$:     MOVL    R8,(SP)                 ; Save new STATE value
                                04D9  1275          ;
                                04D9  1276          ;   If STATE=SERVICE, then make sure that SERVICE is ENABLED.
                                04D9  1277          ;
                                04D9  1278          $GETFLD cri,v,ser               ; See if service functions are enabled
                      0F 50     E9  04E6  1279          BLBC    R0,10$                  ; Branch if parameter not set
                      0C 58     E9  04E9  1280          BLBC    R8,10$                  ; If LBC then "service" state is legal
             50   00000000'8F    D0  04EC  1281          MOVL    #SS$_BADPARAM,R0        ; Assume state is "service"
                      02    6E    D1  04F3  1282          CMPL    (SP),#NMA$C_STATE_SER   ; Is it?
                            78    13  04F6  1283          BEQL    200$                    ; If so then conflicts with <cri,v,ser>
                                04F8  1284          ;
                                04F8  1285          ;   Make sure that the circuit name is valid
                                04F8  1286          ;
                            50    D4  04F8  1287 10$:     CLRL    R0                      ; No default error status
                                04FA  1288          $GETFLD cri,s,vmsnam            ; Check if circuit name is valid
                        66 50     E9  0507  1289          BLBC    R0,200$                 ; If LBC then error
                                050A  1290          ;
                                050A  1291          ;   If the circuit type associated with this CNF is X25, then ship
                                050A  1292          ;   the X.25 specific circuit parameters to PSIACP via a control QIO.
                                050A  1293          ;   If the QIO succeeds, then allow the CNF to be inserted.  In
                                050A  1294          ;   either case, we do not perform any datalink processing.
                                050A  1295          ;
                                050A  1296          $GETFLD cri,l,typ               ; Get circuit type
                          05 50     E9  0517  1297          BLBC    R0,25$                  ; If error, assume not X.25
                      03    58    D1  051A  1298          CMPL    R8,#NMA$C_CIRTY_X25     ; X25 circuit?
                            2A    13  051D  1299          BEQL    28$                     ; Branch if so
                            5A    DD  051F  1300 25$:     PUSHL   R10                     ; Save pointer to current CNF
                      5A    56    D0  0521  1301          MOVL    R6,R10                  ; Point to old version of CNF
                            1E    13  0524  1302          BEQL    26$                     ; Branch if none
                                0526  1303          $GETFLD cri,l,typ               ; Get old circuit type
                          0E 50     E9  0533  1304          BLBC    R0,26$                  ; If error, assume still not X.25
                      03    58    D1  0536  1305          CMPL    R8,#NMA$C_CIRTY_X25     ; Did it used to be X25?
                            09    12  0539  1306          BNEQ    26$                     ; Branch if not
                50    0000'8F   3C  053B  1307          MOVZWL  #SS$_WRITLCK,R0         ; Do not allow type change w/o CLEAR
                            5A 8ED0   0540  1308          POPL    R10                     ; Restore current CNF
```

```
                 05  0543  1309              RSB                             ; Return - do not allow insertion
              5A 8ED0  0544  1310 26$:       POPL    R10                     ; Restore current CNF
              18   11  0547  1311             BRB     50$                     ; and allow Transport to use it
                       0549  1312
              28   10  0549  1313 28$:       BSBB    CRI_TO_PSI              ; Tell it to PSIACP
           22 50   E9  054B  1314             BLBC    R0,200$                 ; Exit if error detected
                       054E  1315             ;
                       054E  1316             ;   For X.25 circuits, do not initialize the circuit if we haven't
                       054E  1317             ;   been granted ownership of it by the network manager.  This
                       054E  1318             ;   prevents the circuit from being used both by native mode X.25
                       054E  1319             ;   users and by Transport.
                       054E  1320             ;
                       054E  1321 29$:       $GETFLD cri,v,DLM               ; X.25 datalink mapping enabled?
           0C 50   E9  055B  1322             BLBC    R0,60$                  ; If not set, do not use this circuit
           09 58   E9  055E  1323             BLBC    R8,60$                  ; If disabled, do not use this circuit
                       0561  1324                                             ; Call Transport to init circuit
                       0561  1325             ;
                       0561  1326             ;   Signal datalink control module that the CRI has been changed
                       0561  1327             ;
        58   6F   D0  0561  1328 50$:       MOVL    (SP),R8                 ; Get STATE value
             FA99'  30  0564  1329             BSBW    NET$DLL_OPR_SET         ; Signal operator event
           06 50   E9  0567  1330             BLBC    R0,200$                 ; If LBC then error
                       056A  1331             ;
                       056A  1332             ;   Start counter timer ticking
                       056A  1333             ;
             FA93'  30  056A  1334 60$:       BSBW    NET$SET_CTR_TIMER       ; Reset automatic counter timer
        50   01   D0  056D  1335             MOVL    #1,R0                   ; Indicate success
              8E   D5  0570  1336 200$:      TSTL    (SP)+                   ; Cleanup the stack
                   05  0572  1337             RSB                             ; Return status in R0
```

```
                                                     L 1
                     0573  1339          .SBTTL  CRI_TO_PSI - Send CRI parameters to PSIACP
                     0573  1340 ;+
                     0573  1341 ; CRI_TO_PSI - Issue SET of CRI parameters to PSIACP
                     0573  1342 ;
                     0573  1343 ; Send the CRI parameters that PSIACP cares about via a ACP control QIO.
                     0573  1344 ;
                     0573  1345 ; Inputs:
                     0573  1346 ;
                     0573  1347 ;       R11 = Address of CNR
                     0573  1348 ;       R10 = Address of CNF
                     0573  1349 ;
                     0573  1350 ; Outputs:
                     0573  1351 ;
                     0573  1352 ;       R0 = Status code
                     0573  1353 ;-
                     0573  1354
                     0573  1355 CRI_TO_PSI:
                     0573  1356          ;
                     0573  1357          ;   Check if required parameters are present
                     0573  1358          ;
 50  0000'8F   3C   0573  1359          MOVZWL  #SS$_INSFARG,R0         ; Set error if parameter not present
                     0578  1360          $GETFLD cri,l_use               ; Check if USAGE parameter specified
       66 50    E9  0585  1361          BLBC    R0,90$                  ; If not, error - must be specified
                     0588  1362          ;
                     0588  1363          ;   If this is an incoming SVC, then make sure that NETACP is
                     0588  1364          ;   declared (to PSI) as being able to accept incoming calls
                     0588  1365          ;   since PSI might not have been up earlier to receive the
                     0588  1366          ;   declaration when the SUBADDRESS parameter was set.
                     0588  1367          ;
       01    58    D1  0588  1368          CMPL    R8,#NMA$C_CIRUS_INC     ; Incoming SVC?
             1C    12  058B  1369          BNEQ    5$                      ; Branch if not
         0C00 8F   BB  058D  1370          PUSHR   #^M<R10,R11>            ; Save registers
5B 00000000'EF   D0  0591  1371          MOVL    NET$GL_CNR_LNI,R11      ; Get LNI root address
5A 00000000'EF   D0  0598  1372          MOVL    NET$GL_PTR_LNI,R10      ; Get LNI CNF address
            FA5E'  30  059F  1373          BSBW    NET$DECLARE_PSI         ; Declare ourselves to accept calls
         0C00 8F   BA  05A2  1374          POPR    #^M<R10,R11>            ; Restore registers
           45 50   E9  05A6  1375          BLBC    R0,90$                  ; Exit if error detected
                     05A9  1376 5$:       ;
                     05A9  1377          ;   If this is a PVC, then go ahead and tell PSI.  If it's a
                     05A9  1378          ;   incoming or outgoing SVC, then don't bother, since PSI
                     05A9  1379          ;   doesn't need to know about them.
                     05A9  1380          ;
       00    58    D1  05A9  1381          CMPL    R8,#NMA$C_CIRUS_PER     ; PVC?
             40    12  05AC  1382          BNEQ    90$                     ; If not, then exit successfully
                     05AE  1383          ;
                     05AE  1384          ;   Send the request to PSI
                     05AE  1385          ;
                     05AE  1386          $CNFFLD cri,s,nam,R9            ; Set field ID of circuit name
           0383    30  05B5  1387          BSBW    SEND_TO_PSI             ; Call co-routine to setup NFB, etc.
         33 50   E9  05B8  1388          BLBC    R0,90$                  ; Branch if error detected
       66    23    90  05BB  1389          MOVB    #NFB$C_FC_SET,NFB$B_FCT(R6)     ; Store function code
55 0000010C'EF   9E  05BE  1390          MOVAB   CRI_PSI_TAB,R5          ; Point to PSI circuit parameters
       59    85    D0  05C5  1391 10$:     MOVL    (R5)+,R9                ; Get next parameter ID
             22    13  05C8  1392          BEQL    50$                     ; Branch if end of table
            FA33'  30  05CA  1393          BSBW    CNF$GET_FIELD           ; Get the parameter value
         F5 50   E9  05CD  1394          BLBC    R0,10$                  ; Skip it if it doesn't exist
       82    59    D0  05D0  1395          MOVL    R9,(R2)+                ; Store into NFB
```

```
                02  10  ED  05D3  1396          CMPZV   #NFB$V_TYP,#NFB$S_TYP,- ; String or binary?
                02  59      05D6  1397                  R9,#NFB$C_TYP_STR
                    05  13  05D8  1398          BEQL    20$                     ; Branch if string
                83  58  D0  05DA  1399          MOVL    R8,(R3)+                ; Store longword into P4 buffer
                    E6  11  05DD  1400          BRB     10$
                83  57  B0  05DF  1401 20$:     MOVW    R7,(R3)+                ; Store string count word
                    34  BB  05E2  1402          PUSHR   #^M<R2,R4,R5>           ; Save registers
            63  68  57  28  05E4  1403          MOVC    R7,(R8),(R3)            ; Store string text
                    34  BA  05E8  1404          POPR    #^M<R2,R4,R5>           ; Restore registers
                    D9  11  05EA  1405          BRB     10$
                    9E  16  05EC  1406 50$:     JSB     @(SP)+                  ; Call SEND_TO_PSI back to issue QIO
                        05  05EE  1407 90$:     RSB                             ; Return with status
```

```
                     05EF  1409          .SBTTL  NET$INSERT_PLI - Pre-insertion processing
                     05EF  1410 ;+
                     05EF  1411 ; NET$INSERT_PLI - Insert PLI entry into database
                     05EF  1412 ;
                     05EF  1413 ; This routine is called to validate a PLI CNF entry before inserting
                     05EF  1414 ; it into the database.
                     05EF  1415 ;
                     05EF  1416 ; Inputs:
                     05EF  1417 ;
                     05EF  1418 ;       R11 = Address of CNR
                     05EF  1419 ;       R10 = Address of CNF
                     05EF  1420 ;       R6 = Address of previous version of CNF (zero if none)
                     05EF  1421 ;
                     05EF  1422 ; Outputs:
                     05EF  1423 ;
                     05EF  1424 ;       R0 = Status code
                     05EF  1425 ;-
                     05EF  1426 NET$INSERT_PLI::
                     05EF  1427          ;
                     05EF  1428          ;   Get new value of STATE parameter.
                     05EF  1429          ;
                     05EF  1430          $GETFLD pli,l,sta               ; Get the operator state
00000000'EF    58    90  05FC  1431          MOVB    R8,PLI_B_STATE          ; Save state value
                     0603  1432          ;
                     0603  1433          ;   If the protocol associated with this CNF is LAPB, then ship
                     0603  1434          ;   the X.25 specific line parameters to PSIACP via a control QIO.
                     0603  1435          ;   If the QIO succeeds, then allow the CNF to be inserted.  In
                     0603  1436          ;   either case, we do not perform any PLVEC or datalink processing.
                     0603  1437          ;
                     0603  1438          $GETFLD pli,l,pro               ; Get protocol type
      12 50    E9    0610  1439          BLBC    R0,25$                  ; If error, assume not X.25
05    58    D1    0613  1440          CMPL    R8,#NMA$C_LINPR_LAPB    ; LAPB protocol?
      0D    12    0616  1441          BNEQ    25$                     ; Branch if not
   02D6     30    0618  1442          BSBW    PLI_TO_PSI              ; If so, send it to PSIACP
   06 50    E9    061B  1443          BLBC    R0,24$                  ; Branch if error detected
    F9DF'   30    061E  1444          BSBW    NET$SET_CTR_TIMER       ; Start/reset automatic counter timer
50    01    D0    0621  1445          MOVL    #1,R0                   ; Set successful
                05  0624  1446 24$:    RSB                             ; Exit with status
      5A    DD    0625  1447 25$:    PUSHL   R10                     ; Save pointer to current CNF
5A    56    D0    0627  1448          MOVL    R6,R10                  ; Point to old version of CNF
      1E    13    062A  1449          BEQL    26$                     ; Branch if none
                  062C  1450          $GETFLD pli,l,pro               ; Get old protocol type
   0E 50    E9    0639  1451          BLBC    R0,26$                  ; If error, assume still not X.25
05    58    D1    063C  1452          CMPL    R8,#NMA$C_LINPR_LAPB    ; Did it used to be LAPB?
      09    12    063F  1453          BNEQ    26$                     ; Branch if not
50  0000'8F    3C  0641  1454          MOVZWL  #SS$_WRITLCK,R0         ; Do not allow protocol change w/o CLEAR
      5A 8ED0     0646  1455          POPL    R10                     ; Restore current CNF
                05  0649  1456          RSB                             ; Return - do not allow insertion
      5A 8ED0     064A  1457 26$:    POPL    R10                     ; Restore current CNF
                  064D  1458          ;
                  064D  1459          ;   Locate the PLVEC index for this PLI.  If there is none then
                  064D  1460          ;   assign and initialize a PLVEC entry for this PLI.
                  064D  1461          ;
  54  12 AA    3C  064D  1462          MOVZWL  CNF$W_ID(R10),R4        ; Get i.d. (PLVEC index) of this entry
      40    12  0651  1463          BNEQ    30$                     ; If NEQ then its valid
   00A9     30  0653  1464          BSBW    ALLOC_PLVEC             ; This is a new PLI, set it up
   38 50    E9  0656  1465          BLBC    R0,29$                  ; If error then simply exit
```

```
                                 0659  1466          SETBIT  PLI_V_NEW,CNF$B_FLG(R10); Indicate PLVEC has been newly assigned
                          01  90 065D  1467          MOVB    #NMA$C_STATE_OFF,-      ; Init to "off" state
              00000000'EF44      065F  1468                      PLVEC$AB_STATE[R4]  ; Setup an undefined state value
                                 0665  1469          ;
                                 0665  1470          ;   If this is an NI driver, then make sure that the executor
                                 0665  1471          ;   type is Phase IV, since Phase III doesn't support
                                 0665  1472          ;   broadcast circuits.
                                 0665  1473          ;
09    00000000'EF44  91          0665  1474          CMPB    PLVEC$AB_DEV[R4],#DEVTRN$C_DEV_UNA ; NI device?
                        24  12   066D  1475          BNEQ    28$                     ; Branch if not
   50   00000000'EF   D0   066F  1476          MOVL    NET$GL_PTR_VCB,R0       ; Get RCB address
        50   008A C0   9A   0676  1477          MOVZBL  RCB$B_ETY(R0),R0        ; Get executor type
                                 067B  1478          $DISPATCH R0,<-                 ; Allow only if set to one of:
                                 067B  1479                  <ADJ$C_PTY_PH4,28$>,-   ; Phase IV routing
                                 067B  1480                  <ADJ$C_PTY_PH4N,28$>,-  ; Phase IV endnode
                                 067B  1481                  <ADJ$C_PTY_AREA,28$>>   ; Phase IV area routing
        50    0000'8F   3C   0685  1482          MOVZWL  #SS$_BADPARAM,R0        ; Else, illegal protocol
                                 068A  1483          $CNFFLD pli,l,pro,R9            ; Indicate parameter in error
                        5B  11   0691  1484 29$:    BRB     100$                    ; Exit with error
                                 0693  1485 28$:    ;
                                 0693  1486          ;   Issue the line control QIO to setup the driver with the control
                                 0693  1487          ;   information.  The IO$_STARTUP and IO$_SHUTDOWN modifiers are
                                 0693  1488          ;   illegal unless there is a state change.
                                 0693  1489          ;
7E    00000000'EF44  9A          0693  1490 30$:    MOVZBL  PLVEC$AB_STATE[R4],-(SP); Save old state value
                          51  D4   069B  1491          CLRL    R1                      ; Clear illegal I/O modifier mask
        00000000'EF   91   069D  1492          CMPB    PLI_B_STATE,-           ; Are old and new state value the same?
              00000000'EF44      06A3  1493                      PLVEC$AB_STATE[R4]  ;
                          05  12   06A9  1494          BNEQ    40$                     ; If NEQ then state change
                0000'8F   B0   06AB  1495          MOVW    #IO$M_STARTUP!-         ; Specify unwanted I/O modifiers
                          51       06AF  1496                   IO$M_SHUTDOWN,R1   ;
        00000000'EF   90   06B0  1497 40$:    MOVB    PLI_B_STATE,-           ; Setup new state value
              00000000'EF44      06B6  1498                      PLVEC$AB_STATE[R4]  ;
                                 06BC  1499          $CNFFLD pli,s,chr,R9            ; Set field i.d.
52    00000000'EF44  3C          06C3  1500          MOVZWL  PLVEC$AW_CHAN[R4],R2    ; Get channel
                        0176  30   06CB  1501          BSBW    NET$SET_QIOW            ; Issue set characteristics QIO
                     59   51   D0   06CE  1502          MOVL    R1,R9                   ; Return 2nd IOSB longword as well
00000000'EF44        8E   F6   06D1  1503          CVTLB   (SP)+,PLVEC$AB_STATE[R4]; Restore original state
                     12 50   E9   06D9  1504          BLBC    R0,100$                 ; If LBC then error
        00000000'EF   90   06DC  1505          MOVB    PLI_B_STATE,-           ; Setup the new state value
              00000000'EF44      06E2  1506                      PLVEC$AB_STATE[R4]  ;
                                 06E8  1507          ;
                                 06E8  1508          ;   Start counter timer ticking
                                 06E8  1509          ;
                     F915'  30   06E8  1510          BSBW    NET$SET_CTR_TIMER       ; Reset automatic counter timer
                  50   01   D0   06EB  1511          MOVL    #1,R0                   ; Set successful
                                 06EE  1512          ;
                                 06EE  1513          ;   If the PLI is not new then simply return the status in R0.  If
                                 06EE  1514          ;   the PLI is new then the newly allocated PLVEC slot must be
                                 06EE  1515          ;   deallocated if R0 indicates an error.
                                 06EE  1516          ;
                        04   E5  06EE  1517 100$:    BBCC    #PLI_V_NEW,-            ; Br unless PLVEC has been newly
                  0B 0B AA       06F0  1518                      CNF$B_FLG(R10),110$ ; assigned
                     08 50   E8  06F3  1519          BLBS    R0,110$                 ; If BS then no clean-up needed
                        50   DD  06F6  1520          PUSHL   R0                      ; Save error code
                     040B   30   06F8  1521          BSBW    DEAL_PLVEC              ; Free the PLVEC cell
                        50 8ED0  06FB  1522          POPL    R0                      ; Restore status
```

```
 05  06FE  1523 110$:   RSB                             ; Return status in R0
```

```
                                06FF  1525          .SBTTL  ALLOC_PLVEC - Setup PLVEC entry for new line
                                06FF  1526 ;+
                                06FF  1527 ; ALLOC_PLVEC - Setup a new PLVEC entry for a new PLI entry
                                06FF  1528 ;
                                06FF  1529 ; Inputs:
                                06FF  1530 ;
                                06FF  1531 ;       R11 = Address of CNR
                                06FF  1532 ;       R10 = Address of CNF
                                06FF  1533 ;
                                06FF  1534 ; Outputs:
                                06FF  1535 ;
                                06FF  1536 ;       R0 = Status code
                                06FF  1537 ;-
                                06FF  1538
                                06FF  1539 ALLOC_PLVEC:                             ; Initialize a new PLVEC entry
     50    0000'8F    3C  06FF  1540          MOVZWL  #SS$_INSFMEM,R0                ; Assume no resources for this entry
54    00000000'EF    9A  0704  1541          MOVZBL  PLVEC$GB_MAX,R4                ; Else no i.d. yet, get max PLX index
   00000000'EF44    95  070B  1542 10$:     TSTB    PLVEC$AB_REFC[R4]              ; Is this cell free?
             06    13  0712  1543          BEQL    20$                            ; If EQL then yes
          F4 54    F5  0714  1544          SOBGTR  R4,10$                         ; Loop -- index 0 is not used
           0113    31  0717  1545 19$:     BRW     100$                           ; Report "resource error"
       12 AA   54    B0  071A  1546 20$:     MOVW    R4,CNF$W_ID(R10)               ; Use PLVEC index as PLI i.d.
   00000000'EF44    96  071E  1547          INCB    PLVEC$AB_REFC[R4]              ; Claim the PLVEC cell
   00000000'EF44    B4  0725  1548          CLRW    PLVEC$AW_CHAN[R4]              ; No channel assigned yet
     50    0000'8F    3C  072C  1549          MOVZWL  #SS$_NOSUCHDEV,R0              ; Assume error
                       0731  1550          $GETFLD pli,s_nam                      ; Else get the line name
          D6 50    E9  073E  1551          BLBC    R0,19$                         ; Branch to 100$ if cannot translate it
                       0741  1552          ;
                       0741  1553          ;   Get the VMS name of the device and the device code.  Assign an
                       0741  1554          ;   I/O channel to the device and squirrel away the device code
                       0741  1555          ;
          5E    17    C2  0741  1556          SUBL    #MAX_C_DEVNAM+8,SP             ; Create scatch space on stack
       53    08 AE    9E  0744  1557          MOVAB   8(SP),R3                       ; Point to scratch space for name text
          6E    53    CE  0748  1558          MNEGL   R3,(SP)                        ; Bias count field of name descriptor
       04 AE    53    D0  074B  1559          MOVL    R3,4(SP)                       ; Enter ptr field of name descriptor
             0C1A    30  074F  1560          BSBW    TRAN_DEVNAM                    ; Translate device name to VMS format
                       0752  1561                                                 ; - advances R3, table entry -> R6
          2E 50    E9  0752  1562          BLBC    R0,40$                         ; If LBC then error
     50    0000'8F    3C  0755  1563          MOVZWL  #SS$_IVDEVNAM,R0               ; Assume error
                57    D5  075A  1564          TSTL    R7                             ; Any unparsed characters?
                25    12  075C  1565          BNEQ    40$                            ; If NEQ yes, name format error
       54    12 AA    3C  075E  1566          MOVZWL  CNF$W_ID(R10),R4               ; Recover R4
          0A A6    90  0762  1567          MOVB    DEVTRN$B_DEV(R6),-             ;
   00000000'EF44        0765  1568                      PLVEC$AB_DEV[R4]           ; Enter device type code
          6E    53    C0  0769  1569          ADDL    R3,(SP)                        ; Complete device name descriptor
          53    5E    D0  076E  1570          MOVL    SP,R3                          ; Point to it
                       0771  1571          $ASSIGN_S -                            ; Assign an I/O channel
                       0771  1572                  CHAN    = PLVEC$AW_CHAN[R4],-
                       0771  1573                  DEVNAM = (R3)
          5E    17    C0  0783  1574 40$:     ADDL    #MAX_C_DEVNAM+8,SP             ; Restore stack
            4D 50    E9  0786  1575          BLBC    R0,49$                         ; If LBC then error
50  00000000'EF44    3C  0789  1576          MOVZWL  PLVEC$AW_CHAN[R4],R0           ; Get I/O channel
     00000000'GF    16  0791  1577          JSB     G^IOC$VERIFYCHAN               ; Return CCB in R1 regardless of error
             61    D0  0797  1578          MOVL    CCB$L_UCB(R1),-                ; Store device UCB
   00000000'EF44        0799  1579                      PLVEC$AL_UCB[R4]
                       079F  1580                                                 ;
          5E    04    C2  079F  1581          SUBL    #4,SP                          ; Use stack for device characteristics
```

```
                53   5E   D0  07A2  1582          MOVL    SP,R3                           ; Point R3 at return buffer
                     00   DD  07A5  1583          PUSHL   #0                              ; Terminate GETDVI item list
          00000031'EF   9F  07A7  1584          PUSHAB  QIOW_W_RETLEN                   ; Point to result length word
                     53   DD  07AD  1585          PUSHL   R3                              ; Point to DEVCHAR return buffer
                     7E   02  B0  07AF  1586          MOVW    #DVI$_DEVCHAR,-(SP)             ; Item code
                     7E   04  B0  07B2  1587          MOVW    #4,-(SP)                        ; Size of DEVCHAR result buffer
                     51   5E   D0  07B5  1588          MOVL    SP,R1                           ; Point to item list
                                  07B8  1589          $GETDVI_S ITMLST=(R1),-                 ; Get device characteristics
                                  07B8  1590                    EFN=#NET$C_EFN_WAIT,-         ; Use synchronous event flag
                                  07B8  1591                    CHAN = PLVEC$AW_CHAN[R4];
                     5E   14  C0  07D3  1592          ADDL    #5*4,SP                         ; Pop item list
                                  07D6  1593                                                  ;
                   55 50  E9  07D6  1594 49$:     BLBC    R0,120$                         ; Branch if error detected
                50   0000'8F  3C  07D9  1595          MOVZWL  #SS$_IVDEVNAM,R0                ; Assume error
                  4C 63   0D  E1  07DE  1596          BBC     #DEV$V_NET,(R3),120$            ; Br if not a network device
                                  07E2  1597 ;
                                  07E2  1598 ; Pre-allocate (and deallocate) the receive buffers for the datalink
                                  07E2  1599 ; driver from the system nonpaged pool.  This is done here before
                                  07E2  1600 ; invoking the driver so that the pool expansion mechanism can be used
                                  07E2  1601 ; in case there isn't enough pool.  It is done here because pool
                                  07E2  1602 ; expansion can't be done on the interrupt stack in the datalink driver.
                                  07E2  1603 ;
                                  07E2  1604          $GETFLD pli,l,bus                       ; Get datalink receive buffer size
                     38 50  E9  07EF  1605          BLBC    R0,70$                          ; If not set, skip it
          55   58  0000004C 8F  C1  07F2  1606          ADDL3   #CXB$C_OVERHEAD,R8,R5           ; Add in datalink's overhead
                                  07FA  1607          $GETFLD pli,l,bfn                       ; Get number of line receive buffers
                   20 50  E9  0807  1608          BLBC    R0,70$                          ; Wasn't defaulted properly - skip it
                      7E   D4  080A  1609          CLRL    -(SP)                           ; Terminate list of allocated blocks
                51    55   D0  080C  1610 50$:     MOVL    R5,R1                           ; Set size of block to allocate
          00000000'EF   16  080F  1611          JSB     NET$ALONPAGED                   ; Allocate from nonpaged pool
                05 50   E9  0815  1612          BLBC    R0,60$                          ; If error, give up
                   52   DD  0818  1613          PUSHL   R2                              ; Save block address
                EF 58   F5  081A  1614          SOBGTR  R8,50$                          ; Loop until all allocated
                   50 8ED0  081D  1615 60$:     POPL    R0                              ; Get next buffer address
                   08   13  0820  1616          BEQL    70$                             ; Branch if all blocks deallocated
          00000000'EF   16  0822  1617          JSB     NET$DEALLOCATE                  ; Deallocate the block
                   F3   11  0828  1618          BRB     60$                             ; Loop until all blocks cleaned up
             50    01   D0  082A  1619 70$:     MOVL    #1,R0                           ; Success
                        05  082D  1620 100$:    RSB                                     ; Return status in R0
                            082E  1621
                            082E  1622 120$:    ;
                            082E  1623          ;  DEASSIGN the channel on error
                            082E  1624          ;
                   50   DD  082E  1625          PUSHL   R0                              ; Save error code
                            0830  1626          $DASSGN_S -                             ; Deassign the I/O channed
                            0830  1627                    CHAN = PLVEC$AW_CHAN[R4];
                   50 8ED0  083F  1628          POPL    R0                              ; Restore error code
                   E9   11  0842  1629          BRB     100$                            ; And exit
```

```
                     0844  1631          .SBTTL  NET$SET_QIOW - Issue datalink SETMODE function
                     0844  1632 ;+
                     0844  1633 ; NET$SET_QIOW - Issue datalink SETMODE function
                     0844  1634 ;
                     0844  1635 ; This routine is called when a line or circuit is to be initialized
                     0844  1636 ; with a SETMODE function.
                     0844  1637 ;
                     0844  1638 ; Inputs:
                     0844  1639 ;
                     0844  1640 ;       R11 = Address of CNR
                     0844  1641 ;       R10 = Address of CNF
                     0844  1642 ;       R9 = Parameter ID for CHR field, which returns DLLQIO buffer
                     0844  1643 ;       R2 = Channel to datalink driver
                     0844  1644 ;
                     0844  1645 ; Outputs:
                     0844  1646 ;
                     0844  1647 ;       R0/R1 = I/O status quadword
                     0844  1648 ;       R9 = Parameter ID qualifying error, if any.
                     0844  1649 ;-
                     0844  1650
                     0844  1651 NET$SET_QIOW::
50   0000'8F   3C    0844  1652          MOVZWL  #SS$_NOSUCHDEV,R0       ; Assume error
       F7B'    30    0849  1653          BSBW    CNF$GET_FIELD           ; Get datalink DLLQIO buffer
     7D 50     E9    084C  1654          BLBC    R0,100$                 ; If LBC then error
   68    51     CA    084F  1655          BICL    R1,DLLQIO$L_FUNC(R8)    ; Mask out illegal I/O modifier bits
     04 A8     D5    0852  1656          TSTL    DLLQIO$L_P1(R8)         ; Any P1 buffer ?
        04     13    0855  1657          BEQL    5$                      ; If EQL no
        58     C0    0857  1658          ADDL    R8,DLLQIO$L_P1(R8)      ; Else turn offset into a pointer
     A8 D5     085B  1659 5$:     TSTL    DLLQIO$L_P2(R8)         ; Any P2 buffer ?
        0C     13    085E  1660          BEQL    10$                     ; If EQL then none
        58     C0    0860  1661          ADDL    R8,DLLQIO$L_P2(R8)      ; Turn offset into a pointer
   08 A8     D0    0864  1662          MOVL    DLLQIO$L_P2(R8),R0      ; Point to P2 buffer descriptor
        58     C0    0868  1663          ADDL    R8,4(R0)                ; Turn offset into a pointer
                     086C  1664 10$:    $QIOW_S -                       ; Set the device charactersitics
                     086C  1665                  EFN  = #NET$C_EFN_WAIT,-
                     086C  1666                  IOSB = QIOW_Q_IOSB,-
                     086C  1667                  CHAN = R2,-
                     086C  1668                  FUNC = DLLQIO$L_FUNC(R8),-
                     086C  1669                  P1   = @DLLQIO$L_P1(R8),-       ; This is a PUSHAL
                     086C  1670                  P2   = DLLQIO$L_P2(R8),-
                     086C  1671                  P3   = DLLQIO$L_P3(R8)
        07 50     E8    0890  1672          BLBS    R0,20$                  ; Branch if queued ok
00000029'EF   50    D0    0893  1673          MOVL    R0,QIOW_Q_IOSB          ; If error, store status into IOSB
59   0000002D'EF    D0    089A  1674 20$:     MOVL    QIOW_Q_IOSB+4,R9        ; Get remainder of IOSB
00000B04 8F   59    D1    08A1  1675          CMPL    R9,#NMA$C_PCLI_PHA      ; "physical address"?
        07     12    08A8  1676          BNEQ    22$                     ; If so,
59   00000488 8F    D0    08AA  1677          MOVL    #NMA$C_PCLI_HWA,R9      ; then map it to "hardware address"
                     08B1  1678                                          ; (so that we see it at the NICE level)
            0A70     30    08B1  1679 22$:     BSBW    NET$CVT_NMA_INT         ; Convert NMA parameter i.d. to an
                     08B4  1680                                          ; internal parameter i.d.
    3FFF 8F    59    B1    08B4  1681          CMPW    R9,#CNVTAB$C_INTRNL     ; Datalink only parameter?
        07     12    08B9  1682          BNEQ    25$                     ; Branch if not
59   0000002D'EF    D0    08BB  1683          MOVL    QIOW_Q_IOSB+4,R9        ; If so, I'd rather see the NMA code
50   00000029'EF    7D    08C2  1684 25$:     MOVQ    QIOW_Q_IOSB,R0          ; Get I/O status
                     08C9  1685          ;
                     08C9  1686          ;   Log the final status into the journal file, if any
                     08C9  1687          ;
```

```
                58    68   3C   08C9  1688           MOVZWL  DLLQIO$L_FUNC(R8),R8     ; Pass I/O function code into journal
00000029'EF     50    7D   08CC  1689 100$:    MOVQ    R0,QIOW_Q_IOSB          ; Save final status
          F72A'       30   08D3  1690           BSBW    NET$JNX_CO              ; Initialize journalling co-routine
             10 50    E9   08D6  1691           BLBC    R0,120$                 ; Branch if not enabled
       81    44 8F    90   08D9  1692           MOVB    #^X44,(R1)+             ; Journal record type = QIOW completion
          81    58    B0   08DD  1693           MOVW    R8,(R1)+                ; I/O function code
81    00000029'EF     7D   08E0  1694           MOVQ    QIOW_Q_IOSB,(R1)+       ; I/O status quadword
                9E    16   08E7  1695           JSB     @(SP)+                  ; Log the journalling record
50    00000029'EF     7D   08E9  1696 120$:    MOVQ    QIOW_Q_IOSB,R0          ; Restore final status
                      05   08F0  1697           RSB
```

```
                                                H  2
                     08F1  1699          .SBTTL  PLI_TO_PSI - Send PLI parameters to PSIACP
                     08F1  1700 ;+
                     08F1  1701 ; PLI_TO_PSI - Issue SET of PLI parameters to PSIACP
                     08F1  1702 ;
                     08F1  1703 ; Send the PLI parameters that PSIACP cares about via a ACP control QIO.
                     08F1  1704 ;
                     08F1  1705 ; Inputs:
                     08F1  1706 ;
                     08F1  1707 ;       R11 = Address of CNR
                     08F1  1708 ;       R10 = Address of CNF
                     08F1  1709 ;
                     08F1  1710 ; Outputs:
                     08F1  1711 ;
                     08F1  1712 ;       R0 = Status code
                     08F1  1713 ;-
                     08F1  1714
                     08F1  1715 PLI_TO_PSI:
55 0000012C'EF   9E  08F1  1716          MOVAB   PSI_PLI_CLR_TAB,R5      ; Get table of params to clear
        FA65     30  08F8  1717          BSBW    CLEAR_VOLATILE          ; Clear volatile parameters
                     08FB  1718          $CNFFLD pli,s,nam,R9            ; Set field ID of line name
          37     10  0902  1719          BSBB    SEND_TO_PSI             ; Call co-routine to setup NFB, etc.
       33 50     E9  0904  1720          BLBC    R0,90$                  ; Branch if error detected
    66    23     90  0907  1721          MOVB    #NFB$C_FC_SET,NFB$B_FCT(R6)     ; Store function code
55 00000134'EF   9E  090A  1722          MOVAB   PLI_PSI_TAB,R5          ; Point to PSI line parameters
    59    85     D0  0911  1723 10$:     MOVL    (R5)+,R9                ; Get next parameter ID
          22     13  0914  1724          BEQL    50$                     ; Branch if end of table
      F6E7'      30  0916  1725          BSBW    CNF$GET_FIELD           ; Get the parameter value
     F5 50     E9  0919  1726          BLBC    R0,10$                  ; Skip it if it doesn't exist
    82    59     D0  091C  1727          MOVL    R9,(R2)+                ; Store into NFB
    02    10     ED  091F  1728          CMPZV   #NFB$V_TYP,#NFB$S_TYP,- ; String or binary?
    02    59         0922  1729                  R9,#NFB$C_TYP_STR
          05     13  0924  1730          BEQL    20$                     ; Branch if string
    83    58     D0  0926  1731          MOVL    R8,(R3)+                ; Store longword into P4 buffer
          E6     11  0929  1732          BRB     10$
    83    57     B0  092B  1733 20$:     MOVW    R7,(R3)+                ; Store string count word
          34     BB  092E  1734          PUSHR   #^M<R2,R4,R5>           ; Save registers
63  68    57     28  0930  1735          MOVC    R7,(R8),(R3)            ; Store string text
          34     BA  0934  1736          POPR    #^M<R2,R4,R5>           ; Restore registers
          D9     11  0936  1737          BRB     10$
          9E     16  0938  1738 50$:     JSB     @(SP)+                  ; Call SEND_TO_PSI back to issue QIO
                 05  093A  1739 90$:     RSB                             ; Return with status
```

```
                        .SBTTL  SEND_TO_PSI - Send control QIO to PSIACP
                ;+
                ; SEND_TO_PSI - Send ACP control QIO to PSIACP
                ;
                ; This routine is called to transmit an ACP control function to PSIACP
                ; on behalf on a local CNF, such as a LAPB line.  CNFs like these are
                ; "known" about by both NETACP and PSIACP, NETACP in order to perform
                ; network management functions, and PSIACP in order to manage the line.
                ;
                ; Inputs:
                ;
                ;       R11 = Address of CNR
                ;       R10 = Address of CNF
                ;       R9 = Address of parameter ID describing entity name
                ;
                ; Outputs:
                ;
                ;       R0 = Status code
                ;
                ;       If the function code is NFB$C_FC_SHOW or NFB$C_FC_ZERCOU, then:
                ;
                ;       R7/R8 = Descriptor of result P4 buffer
                ;       R9 = Address of scratch storage, which should be deallocated
                ;               when processing of the P4 buffer is completed.
                ;
                ;       R1-R6 are destroyed.
                ;
                ; This routine makes a co-routine call back to the caller after setting
                ; up the NFB and associated control QIO buffers, so that the caller can
                ; fill in the essential fields.  When the caller returns via co-routine
                ; call to this routine, the control QIO is issued to PSIACP.
                ;
                ; On input to co-routine:
                ;
                ;       R6 = Address of NFB
                ;       R3 = Address of P4 buffer
                ;       R2 = Address of NFB$L_FLDID list
                ;
                ; On return from co-routine:
                ;
                ;       R3 = Updated pointer to end of P4 buffer
                ;       R2 = Updated pointer to end of FLDID list
                ;
                ;       R0-R1,R4-R5,R7-R9 may be destroyed by co-routine.
                ;-

000000B8        NFB_SIZE = NFB$C_LENGTH + 30*4          ; allow for all PSI field IDs
00000066        P2BUF_SIZE = 4+2+DEVNAM_C_SIZ+NFB$C_CTX_SIZE ; P2 holds CNF count and entity ID
                                                        ; plus context area
00000200        P4BUF_SIZE = 512                        ; P4 holds all parameter values

                        ASSUME  PLI_PSI_SIZ LE 30*4     ; Must be enough room for PLI params

                SEND_TO_PSI:
51   032A 8F    3C      MOVZWL  #12+NFB_SIZE+P2BUF_SIZE+P4BUF_SIZE,R1 ; Set size of block overhead
                                                        ; plus NFB, P2 and P4 buffer sizes
 00000000'EF    16      JSB     NET$ALLOCATE            ; Allocate the storage
```

```
                    01 50    E8  0946  1798          BLBS    R0,5$                           ; Branch if successful
                             05  0949  1799          RSB                                     ; Else, exit with error status
                       52    DD  094A  1800 5$:      PUSHL   R2                              ; Save address of block
                 56    0C A2   9E  094C  1801          MOVAB   12(R2),R6                       ; Point to scratch NFB
        66 10 C0    6E    00   2C  0950  1802          MOVC5   #0,(SP),#0,#NFB$C_LENGTH,(R6)   ; Zero it
              02 A6    0A AB   90  0956  1803          MOVB    CNR$B_TYPE(R11),NFB$B_DATABASE(R6) ; Store database ID
                 01 A6    04   90  095B  1804          MOVB    #NFB$M_NOCTX,NFB$B_FLAGS(R6)    ; Indicate one-shot request
                 03 A6    00   90  095F  1805          MOVB    #NFB$C_OP_EQL,NFB$B_OPER(R6)    ; Search for key EQL
                 04 A6    59   D0  0963  1806          MOVL    R9,NFB$L_SRCH_KEY(R6)           ; Search for specific entry
                                   0967  1807                                                  ; by it's network management name
                       F696'      30  0967  1808          BSBW    CNF$GET_FIELD               ; Get descriptor of name
                    03 50         E8  096A  1809          BLBS    R0,10$                      ; Branch if ok
                     00E8         31  096D  1810          BRW     110$                        ; Abort if unable to get entity name
              00BC C6    57       B0  0970  1811 10$:     MOVW    R7,NFB_SIZE+4(R6)           ; Store size of string
           00BE C6    68    57     28  0975  1812          MOVC    R7,(R8),NFB_SIZE+6(R6)      ; Copy into buffer
                       83          B4  097B  1813          CLRW    (R3)+                       ; Make context a null string
                 52    10 A6       9E  097D  1814          MOVAB   NFB$L_FLDID(R6),R2          ; Point to NFB field ID list
              53    011E C6       9E  0981  1815          MOVAB   NFB_SIZE+P2BUF_SIZE(R6),R3 ; Get address of P4 buffer
                    50   01       D0  0986  1816          MOVL    #1,R0                       ; Indicate co-routine init. successful
                                   0989  1817 ;
                                   0989  1818 ; Call the caller back so that he can fill in the NFB and P4 buffers.
                                   0989  1819 ;
                                   0989  1820          SWAP    (SP),4(SP)                  ; Swap block pointer & return address
                                   0995  1821                                              ; so that caller's address is on top
                          9E   16  0995  1822          JSB     @(SP)+                      ; Fill in the NFB and P4 buffers
                                   0997  1823          SWAP    (SP),4(SP)                  ; Swap return address & block pointer
                                   09A3  1824 ;
                                   09A3  1825 ; Terminate the NFB field ID list
                                   09A3  1826 ;
                                   09A3  1827          ASSUME  NFB$C_ENDOFLIST EQ 0
                          82   D4  09A3  1828          CLRL    (R2)+                       ; Terminate NFB parameter ID list
                                   09A5  1829 ;
                                   09A5  1830 ; Construct descriptors of NFB, P2 and P4 buffers for QIO
                                   09A5  1831 ;
                    7E   6E   0C   C1  09A5  1832          ADDL3   #12,(SP),-(SP)              ; Retreive starting address of NFB
                    7E   52   6E   C3  09A9  1833          SUBL3   (SP),R2,-(SP)               ; Compute length of NFB
        7E 04 AE    0000011E 8F    C1  09AD  1834          ADDL3   #NFB_SIZE+P2BUF_SIZE,4(SP),-(SP) ; Construct descriptor of P4
                    7E   53   6E   C3  09B6  1835          SUBL3   (SP),R3,-(SP)               ; Compute length of P4 buffer
        51 0C AE    000000B8 8F    C1  09BA  1836          ADDL3   #NFB_SIZE,12(SP),R1         ; Get address of P2 buffer
                     50   04 A1    3C  09C3  1837          MOVZWL  4(R1),R0                    ; Pick up length of entity name
                 50  00000046 8F    C0  09C7  1838          ADDL    #4+2+NFB$C_CTX_SIZE,R0      ; Add 4 bytes for CNF count
                                   09CE  1839                                                  ; plus 2 bytes for entity string size
                                   09CE  1840                                                  ; and context area size
                       7E   50   7D  09CE  1841          MOVQ    R0,-(SP)                    ; Push P2 descrptor onto stack
                       52   5E   D0  09D1  1842          MOVL    SP,R2                       ; Point to P2/P4/NFB descriptors
                     53   08 A2   9E  09D4  1843          MOVAB   8(R2),R3                    ; Point to P4 descriptor directly
                                   09D8  1844 ;
                                   09D8  1845 ; Send database request to PSIACP
                                   09D8  1846 ;
                    00000000'EF    B5  09D8  1847          TSTW    NET$GW_X25_CHAN             ; Is there an active channel?
                             06    12  09DE  1848          BNEQ    60$                         ; If NEQL then yes
                          F61D'    30  09E0  1849          BSBW    NET$GET_X25_CHAN            ; Assign channel, get PSI mutex
                        3C 50      E9  09E3  1850          BLBC    R0,100$                     ; If LBC then error
                                   09E6  1851 60$:     $QIOW_S FUNC = #IO$_ACPCONTROL,-    ; Re-issue QIO
                                   09E6  1852                 EFN  = #NET$C_EFN_WAIT,-    ; event flag for synchronous calls
                                   09E6  1853                 CHAN = NET$GW_X25_CHAN,-
                                   09E6  1854                 IOSB = QIOW_Q_IOSB,-        ; Scratch quadword buffer
```

```
                                09E6  1855                    P1    = 8(R3),-           ; Address of NFB descriptor
                                09E6  1856                    P2    = R2,-              ; Address of P2 buffer desciptor
                                09E6  1857                    P3    = #QIOW_W_RETLEN,-  ; Address of return length word
                                09E6  1858                    P4    = R3                ; Address of P4 buffer
                     0D 50   E9 0A12  1859            BLBC    R0,100$                   ; If LBC then error
50   00000029'EF        7D   0A15  1860            MOVQ    QIOW_Q_IOSB,R0              ; Setup IOSB image
                        03 50   E8 0A1C  1861            BLBS    R0,100$                   ; Branch if successful
                     59   51   D0 0A1F  1862            MOVL    R1,R9                     ; Set error qualifier
                     5E   18   C0 0A22  1863 100$:     ADDL    #3*8,SP                   ; Deallocate descriptors
                               0A25  1864
00000029'EF    50        7D   0A25  1865            MOVQ    R0,QIOW_Q_IOSB              ; Save final status
                  F5D1'     30   0A2C  1866            BSBW    NET$JNX_CO                ; Initialize journalling co-routine
                     12 50   E9 0A2F  1867            BLBC    R0,105$                   ; Branch if not enabled
               81    44 8F   90 0A32  1868            MOVB    #^X44,(R1)+               ; Journal record type = QIOW completion
            81    0000'8F   B0 0A36  1869            MOVW    #IO$_ACPCONTROL,(R1)+     ; I/O function code
81   00000029'EF        7D   0A3B  1870            MOVQ    QIOW_Q_IOSB,(R1)+         ; I/O status quadword
                        9E   16   0A42  1871            JSB     @(SP)+                    ; Log the journalling record
50   00000029'EF        7D   0A44  1872 105$:     MOVQ    QIOW_Q_IOSB,R0              ; Restore final status
                               0A4B  1873
                     0A 50   E9 0A4B  1874            BLBC    R0,110$                   ; Branch if error detected
                  22   66   91 0A4E  1875            CMPB    NFB$B_FCT(R6),#NFB$C_FC_SHOW    ; SHOW function?
                        15   13   0A51  1876            BEQL    150$                      ; Branch if so
                  25   66   91 0A53  1877            CMPB    NFB$B_FCT(R6),#NFB$C_FC_ZERCOU  ; READ & ZERO function?
                        10   13   0A56  1878            BEQL    150$                      ; Branch if so
                  51   50   D0 0A58  1879 110$:     MOVL    R0,R1                     ; Save final status
                     50 8ED0   0A5B  1880            POPL    R0                        ; Retreive address of storage
               00000000'EF        16   0A5E  1881            JSB     NET$DEALLOCATE            ; Deallocate scratch storage
                     50   51   D0 0A64  1882            MOVL    R1,R0                     ; Restore final status
                        05   0A67  1883            RSB                               ; Done
                               0A68  1884
                               0A68  1885 ;
                               0A68  1886 ; The function is either SHOW or READ/ZERO.  Since the caller will most
                               0A68  1887 ; likely want to process the returned P4 buffer from the QIO, we return
                               0A68  1888 ; from the routine with a desciptor of the returned buffer, and the
                               0A68  1889 ; address of our scratch storage, which the caller will deallocate.
                               0A68  1890 ;
                               0A68  1891
                  59 8ED0   0A68  1892 150$:     POPL    R9                        ; Return scratch pointer in R9
57   00000031'EF        3C   0A6B  1893            MOVZWL  QIOW_W_RETLEN,R7          ; Return descriptor of P4 buffer
   58    012A C9        9E   0A72  1894            MOVAB   12+NFB_SIZE+P2BUF_SIZE(R9),R8
                        05   0A77  1895            RSB                               ; Exit successfully
```

```
                     0A78  1897          .SBTTL  NET$DELETE_CRI - Pre-delete processing
                     0A78  1898 ;+
                     0A78  1899 ; NET$DELETE_CRI - Special processing before a CRI entry is marked for delete
                     0A78  1900 ;
                     0A78  1901 ; This routine is called to perform any special action that may need to be
                     0A78  1902 ; taken before marking a CNF for delete.
                     0A78  1903 ;
                     0A78  1904 ; INPUTS:        R11     CNR pointer
                     0A78  1905 ;                R10     CNF pointer
                     0A78  1906 ;
                     0A78  1907 ; OUTPUTS:       R11,R10 Preserved
                     0A78  1908 ;                R0      LBS if successful
                     0A78  1909 ;                        LBC if CNF should not be marked for delete
                     0A78  1910 ;
                     0A78  1911 ;                All registers may be destroyed.
                     0A78  1912 ;-
                     0A78  1913
                     0A78  1914 NET$DELETE_CRI::
                     0A78  1915          ;
                     0A78  1916          ; If this is an X25 circuit, ask PSIACP if it's ok to delete CNF
                     0A78  1917          ;
                     0A78  1918          $GETFLD cri,l,typ                   ; Get circuit type parameter value
   31 50    E9       0A85  1919          BLBC    R0,10$                      ; If not specified, skip it
03    58    D1       0A88  1920          CMPL    R8,#NMA$C_CIRTY_X25         ; X25 circuit?
      2C    12       0A8B  1921          BNEQ    10$                         ; Branch if not
                     0A8D  1922          $GETFLD cri,l,use                   ; Get X.25 usage
   1C 50    E9       0A9A  1923          BLBC    R0,10$                      ; If not specified, skip it
00    58    D1       0A9D  1924          CMPL    R8,#NMA$C_CIRUS_PER         ; PVC?
      17    12       0AA0  1925          BNEQ    10$                         ; Only ask PSI about PVCs
                     0AA2  1926          $CNFFLD cri,s,nam,R9                ; Set field ID of circuit name
    FE8F    30       0AA9  1927          BSBW    SEND_TO_PSI                 ; Call co-routne to setup NFB, P4, etc.
   13 50    E9       0AAC  1928          BLBC    R0,90$                      ; Branch if error detected
66    21    90       0AAF  1929          MOVB    #NFB$C_FC_DELETE,NFB$B_FCT(R6) ; Set function code
      9E    16       0AB2  1930          JSB     @(SP)+                      ; Call SEND_TO_PSI to issue QIO
   0B 50    E9       0AB4  1931          BLBC    R0,90$                      ; Branch if error detected
      06    11       0AB7  1932          BRB     50$                         ; Allow delete
                     0AB9  1933          ;
                     0AB9  1934          ;   For all circuits, if the LPD is no longer present (as a result
                     0AB9  1935          ;   of an ON -> OFF state transition), then allow the delete.
                     0AB9  1936          ;
    F544'    30       0AB9  1937 10$:     BSBW    NET$LOCATE_LPD              ; Locate the LPD
   04 50    E8       0ABC  1938          BLBS    R0,80$                      ; If found, then do not allow delete
50    01    D0       0ABF  1939 50$:     MOVL    #1,R0                       ; Allow delete
                     05    0AC2  1940 90$:     RSB
                     0AC3  1941
      50    D4       0AC3  1942 80$:     CLRL    R0                          ; Do not allow delete
                     05    0AC5  1943          RSB
```

```
                     0AC6  1945          .SBTTL  NET$DELETE_PLI - Pre-delete processing
                     0AC6  1946 ;+
                     0AC6  1947 ; NET$DELETE_PLI - Special processing before a PLI entry is marked for delete
                     0AC6  1948 ;
                     0AC6  1949 ; This routine is called to perform any special action that may need to be
                     0AC6  1950 ; taken before marking a CNF for delete.
                     0AC6  1951 ;
                     0AC6  1952 ; INPUTS:        R11     CNR pointer
                     0AC6  1953 ;                R10     CNF pointer
                     0AC6  1954 ;
                     0AC6  1955 ; OUTPUTS:       R11,R10 Preserved
                     0AC6  1956 ;                R0      LBS if successful
                     0AC6  1957 ;                        LBC if CNF should not be marked for delete
                     0AC6  1958 ;
                     0AC6  1959 ;                All registers may be destroyed.
                     0AC6  1960 ;-
                     0AC6  1961
                     0AC6  1962 NET$DELETE_PLI::
                     0AC6  1963          $GETFLD pli,l,sta                   ; Get the state value
      08 50    E9    0AD3  1964          BLBC    R0,10$                      ; If LBC then no need to check state
58    01    D1      0AD6  1965          CMPL    #NMA$C_STATE_OFF,R8         ; Is the line turned off?
      03    13      0AD9  1966          BEQL    10$                         ; If NEQ then cannot delete PLI
      50    D4      0ADB  1967          CLRL    R0                          ; Say "cannot delete PLI"
            05      0ADD  1968          RSB                                 ; Done
                    0ADE  1969          ;
                    0ADE  1970          ; If this is a LAPB line, ask PSIACP if it's ok to delete CNF
                    0ADE  1971          ;
                    0ADE  1972 10$:     $GETFLD pli,l,pro                   ; Get protocol parameter value
      18 50    E9    0AEB  1973          BLBC    R0,DEAL_PLVEC               ; If not specified, skip it
05    58    D1      0AEE  1974          CMPL    R8,#NMA$C_LINPR_LAPB        ; LAPB line?
      13    12      0AF1  1975          BNEQ    DEAL_PLVEC                  ; Branch if not
                    0AF3  1976          $CNFFLD pli,s,nam,R9                ; Set field ID of line name
   FE3E     30      0AFA  1977          BSBW    SEND_TO_PSI                 ; Call co-routne to setup NFB, P4, etc.
   05 50    E9      0AFD  1978          BLBC    R0,90$                      ; Branch if error detected
66    21    90      0B00  1979          MOVB    #NFB$C_FC_DELETE,NFB$B_FCT(R6) ; Set function code
      9E    16      0B03  1980          JSB     @(SP)+                      ; Call SEND_TO_PSI to issue QIO
            05      0B05  1981 90$:     RSB                                 ; Return with status.
                    0B06  1982
                    0B06  1983 DEAL_PLVEC:                                  ; Deallocate PLVEC cell
54    12 AA    3C    0B06  1984          MOVZWL  CNF$W_ID(R10),R4            ; Get the PLVEC index
      1D    13      0B0A  1985          BEQL    10$                         ; If EQL then done
00000000'EF44    94  0B0C  1986          CLRB    PLVEC$AB_REFC[R4]           ; No longer any references to this slot
00000000'EF44    D4  0B13  1987          CLRL    PLVEC$AL_UCB[R4]            ; No UCB
                    0B1A  1988          $DASSGN_S -                         ; Deassign the I/O channed
                    0B1A  1989                  CHAN = PLVEC$AW_CHAN[R4];
                    0B29  1990                                              ; Ignore errors
50    00'   D0      0B29  1991 10$:     MOVL    S^#SS$_NORMAL,R0            ; Indicate success
            05      0B2C  1992          RSB
```

```
                                0B2D  1994          .SBTTL  NET$REMOVE_xxx - Pre-remcve processing
                                0B2D  1995 ;+
                                0B2D  1996 ; NET$REMOVE_CRI - Processing after CRI block has been removed.
                                0B2D  1997 ; NET$REMOVE_PLI - Processing after PLI block has been removed.
                                0B2D  1998 ;
                                0B2D  1999 ; This routine is called to perform special processing after a CNF block has
                                0B2D  2000 ; been removed from the database.  On return, the block is deallocated.
                                0B2D  2001 ;
                                0B2D  2002 ; INPUTS:        R11     CNR pointer
                                0B2D  2003 ;                R0      CNF pointer
                                0B2D  2004 ;
                                0B2D  2005 ; OUTPUTS:       All registers are preserved.
                                0B2D  2006 ;-
                                0B2D  2007 NET$REMOVE_CRI::                             ; Remove Circuit CNF action routine
                                0B2D  2008 NET$REMOVE_PLI::                             ; Remove Line CNF action routine
                      00000000'EF 16  0B2D  2009          JSB     NET$REMOVE_DEF      ; Let default routine do the work
                                05  0B33  2010          RSB
```

```
                        0B34  2012          .SBTTL  CRI parameter action routines
                        0B34  2013 ;+
                        0B34  2014 ; NET$CRI_V_LCK -  Get status of conditionally writeable fields
                        0B34  2015 ;
                        0B34  2016 ; NET$CRI_L_SUB -  Get circuit's substate
                        0B34  2017 ; NET$CRI_L_DRT -  Get broadcast circuit's designated router
                        0B34  2018 ;
                        0B34  2019 ; NET$CRI_L_PNA -  Get partner's node address
                        0B34  2020 ; NET$CRI_L_BLO -  Get partner's receive block size
                        0B34  2021 ; NET$CRI_L_LIT -  Get partner's listen timer
                        0B34  2022 ;
                        0B34  2023 ; NET$CRI_S_COL -  Get circuit's collating value
                        0B34  2024 ; NET$CRI_S_PNN -  Get partner's node name
                        0B34  2025 ; NET$CRI_S_LOO -  Get line's loopback name
                        0B34  2026 ; NET$CRI_S_CNT -  Get (optionally clear) circuit counters
                        0B34  2027 ; NET$CRI_S_VMSNAM Get VMS name of associated device
                        0B34  2028 ; NET$CRI_S_CHR -  Get circuit's characteristics buffer
                        0B34  2029 ;
                        0B34  2030 ; INPUTS:       R11     CRI CNR address
                        0B34  2031 ;               R10     CRI CNF address
                        0B34  2032 ;               R9      FLD i.d. of field being read
                        0B34  2033 ;               R1      Scratch
                        0B34  2034 ;               R0      Scratch
                        0B34  2035 ;
                        0B34  2036 ; OUTPUTS:      R1      Address of string descriptor or address of field value
                        0B34  2037 ;               R0      Low bit set if R1 is valid
                        0B34  2038 ;                       Low bit clear otherwise
                        0B34  2039 ;
                        0B34  2040 ;               All other register values are preserved.
                        0B34  2041 ;
                        0B34  2042 ;-
                        0B34  2043
                        0B34  2044 NET$CRI_V_LCK::                          ; Get status of cond. writeable fields
     F4C9'   30         0B34  2045          BSBW    NET$LOCATE_LPD          ; Locate the LPD
   07 50     E9         0B37  2046          BLBC    R0,10$                  ; If LBC no LPD, not locked
   1B A6     95         0B3A  2047          TSTB    LPD$B_ASTCNT(R6)        ; Any I/O outstanding?
      02     12         0B3D  2048          BNEQ    10$                     ; If NEQ yes, locked
      50     D4         0B3F  2049          CLRL    R0                      ; Else not locked
51    50     D0         0B41  2050 10$:     MOVL    R0,R1                   ; Return as field value
50    00'    D0         0B44  2051          MOVL    S^#SS$_NORMAL,R0        ; Success
             05         0B47  2052          RSB
                        0B48  2053
                        0B48  2054
                        0B48  2055 NET$CRI_L_SUB::                          ; Get circuit's substate
     F4B5'   30         0B48  2056          BSBW    NET$LOCATE_LPD          ; Locate the LPD
   0E 50     E9         0B4B  2057          BLBC    R0,10$                  ; If LBC, no LPD or substate
      50     D4         0B4E  2058          CLRL    R0                      ; Assume in "run" state
      04     E0         0B50  2059          BBS     #LPD$V_RUN,-            ; If BS then no sub-state
07 22 A6                0B52  2060                      LPD$W_STS(R6),10$   ;
51    27 A6     9A      0B55  2061          MOVZBL  LPD$B_SUB_STA(R6),R1    ; Get sub-state
50    00'       D0      0B59  2062          MOVL    S^#SS$_NORMAL,R0        ; Set status
             05         0B5C  2063 10$:     RSB                             ; Return to co-routine
                        0B5D  2064
                        0B5D  2065 NET$CRI_L_DRT::                          ; Get broadcast circuit's designated router
     F4A0'   30         0B5D  2066          BSBW    NET$LOCATE_LPD          ; Locate the LPD
   25 50     E9         0B60  2067          BLBC    R0,90$                  ; If not found, return "not set"
      50     D4         0B63  2068          CLRL    R0                      ; Assume not a valid address
```

```
                1E 22 A6    0A   E1  0B65  2069          BBC     #LPD$V_BC,LPD$W_STS(R6),90$ ; No DRT if not broadcast circuit
                   51    2C A6   3C  0B6A  2070          MOVZWL  LPD$W_DRT(R6),RT            ; Get designated router ADJ index
                            18   13  0B6E  2071          BEQL    90$                         ; If 0, then DRT "not set"
             54  00000000'EF    D0  0B70  2072          MOVL    NET$GL_PTR_VCB,R4           ; Get RCB address
                51    2C B441   D0  0B77  2073          MOVL    @RCB$L_PTR_ADJ(R4)[R1],R1   ; Get ADJ address
                   51    04 A1   3C  0B7C  2074          MOVZWL  ADJ$W_PNA(R1),R1            ; Get DRT node address
                            06   13  0B80  2075          BEQL    90$                         ; If 0, then DRT "not set"
                      F47B'      30  0B82  2076          BSBW    SUPPRESS_AREA               ; Suppress area, if necessary
                         50   00' D0  0B85  2077          MOVL    S^#SS$_NORMAL,R0            ; Set status
                              05   0B88  2078 90$:        RSB                                 ; Exit with status/value
                                   0B89  2079
                                   0B89  2080 NET$CRI_L_BLO::                               ; Get partner's receive block size
                      F474'      30  0B89  2081          BSBW    NET$LOCATE_LPD              ; Locate the LPD
                         1E 50   E9  0B8C  2082          BLBC    R0,10$                      ; If LBC, no LPD or partner node
                            50   D4  0B8F  2083          CLRL    R0                          ; Assume not in "run" state
                            04   E1  0B91  2084          BBC     #LPD$V_RUN,-                ; If BS then no block size
                   17 22 A6         0B93  2085                      LPD$W_STS(R6),10$       ;
                   57    20 A6   9A  0B96  2086          MOVZBL  LPD$B_PTH_INX(R6),R7        ; Get LPD index
             54  00000000'EF    D0  0B9A  2087          MOVL    NET$GL_PTR_VCB,R4           ; Get RCB address
                57    2C B447   D0  0BA1  2088          MOVL    @RCB$L_PTR_ADJ(R4)[R7],R7   ; Get ADJ address
                   51    06 A7   3C  0BA6  2089          MOVZWL  ADJ$W_BUFSIZ(R7),R1         ; Get partner's node address
                         50   00' D0  0BAA  2090          MOVL    S^#SS$_NORMAL,R0            ; Set status
                              05   0BAD  2091 10$:        RSB                                 ; Return to co-routine
                                   0BAE  2092
                                   0BAE  2093 NET$CRI_L_LIT::                               ; Get partner's listen timer
                      F44F'      30  0BAE  2094          BSBW    NET$LOCATE_LPD              ; Locate the LPD
                         23 50   E9  0BB1  2095          BLBC    R0,10$                      ; If LBC, no LPD or partner node
                            50   D4  0BB4  2096          CLRL    R0                          ; Assume not in "run" state
                            04   E1  0BB6  2097          BBC     #LPD$V_RUN,-                ; If BS then no listen timer
                   1C 22 A6         0BB8  2098                      LPD$W_STS(R6),10$       ;
                            0A   E0  0BBB  2099          BBS     #LPD$V_BC,-                 ; If a broadcast circuit,
                   17 22 A6         0BBD  2100                      LPD$W_STS(R6),10$       ; then no listen timer
                   57    20 A6   9A  0BC0  2101          MOVZBL  LPD$B_PTH_INX(R6),R7        ; Get LPD index
             54  00000000'EF    D0  0BC4  2102          MOVL    NET$GL_PTR_VCB,R4           ; Get RCB address
                57    2C B447   D0  0BCB  2103          MOVL    @RCB$L_PTR_ADJ(R4)[R7],R7   ; Get ADJ address
                   51    08 A7   3C  0BD0  2104          MOVZWL  ADJ$W_INT_LSN(R7),R1        ; Get partner's listen timer
                         50   00' D0  0BD4  2105          MOVL    S^#SS$_NORMAL,R0            ; Set status
                              05   0BD7  2106 10$:        RSB                                 ; Return to co-routine
                                   0BD8  2107
                                   0BD8  2108 NET$CRI_L_PNA::                               ; Get partner's node address
                      F425'      30  0BD8  2109          BSBW    NET$LOCATE_LPD              ; Locate the LPD
                         26 50   E9  0BDB  2110          BLBC    R0,10$                      ; If LBC, no LPD or partner node
                            50   D4  0BDE  2111          CLRL    R0                          ; Assume not in "run" state
                            04   E1  0BE0  2112          BBC     #LPD$V_RUN,-                ; If BC then no partner node
                   1F 22 A6         0BE2  2113                      LPD$W_STS(R6),10$       ;
                            0A   E0  0BE5  2114          BBS     #LPD$V_BC,-                 ; If a broadcast circuit,
                   1A 22 A6         0BE7  2115                      LPD$W_STS(R6),10$       ; then no block size
                   57    20 A6   9A  0BEA  2116          MOVZBL  LPD$B_PTH_INX(R6),R7        ; Get LPD index
             54  00000000'EF    D0  0BEE  2117          MOVL    NET$GL_PTR_VCB,R4           ; Get RCB address
                57    2C B447   D0  0BF5  2118          MOVL    @RCB$L_PTR_ADJ(R4)[R7],R7   ; Get ADJ address
                   51    04 A7   3C  0BFA  2119          MOVZWL  ADJ$W_PNA(R7),R1            ; Get partner's node address
                      F3FF'      30  0BFE  2120          BSBW    SUPPRESS_AREA               ; Suppress area, if necessary
                         50   00' D0  0C01  2121          MOVL    S^#SS$_NORMAL,R0            ; Set status
                              05   0C04  2122 10$:        RSB                                 ; Return to co-routine
                                   0C05  2123
                                   0C05  2124 NET$CRI_L_MST::                               ; Get maintenance mode state
                      F3F8'      30  0C05  2125          BSBW    NET$LOCATE_LPD              ; Locate the LPD
```

```
                0E 50   E9  0C08  2126          BLBC    R0,20$                          ; If LBC, no LPD or partner node
              51   00   3C  0C0B  2127          MOVZWL  #NMA$C_STATE_ON,R1              ; Assume state is "ON"
                   02   E0  0C0E  2128          BBS     #LPD$V_DLE,-                    ; If BS then maintenance state is "ON"
            03 22 A6        0C10  2129                      LPD$W_STS(R6),10$           ;
              51   01   3C  0C13  2130          MOVZWL  #NMA$C_STATE_OFF,R1             ; Else state is "OFF"
              50   00'  D0  0C16  2131 10$:     MOVL    S^#SS$_NORMAL,R0                ; Set status
                   05   0C19  2132 20$:     RSB                                     ; Return to co-routine
                        0C1A  2133
                        0C1A  2134
                        0C1A  2135 NET$CRI_S_COL::                                  ; Get circuit's collating value
                        0C1A  2136          $GETFLD cri,s,nam                       ; Get circuit's name
                09 50   E9  0C27  2137          BLBC    R0,10$                          ; Branch if not present
          63    68   57   28  0C2A  2138          MOVC3   R7,(R8),(R3)                    ; Move the name
          50    0000'8F   3C  0C2E  2139          MOVZWL  #SS$_NORMAL,R0                  ; Indicate success
                   05   0C33  2140 10$:     RSB                                     ; Return status in R0
                        0C34  2141
                        0C34  2142 NET$CRI_S_PNN::                                  ; Get partner's node name
                F3C9'   30  0C34  2143          BSBW    NET$LOCATE_LPD                  ; Locate the LPD
              46 50     E9  0C37  2144          BLBC    R0,20$                          ; If LBC, no LPD or partner node
                 50     D4  0C3A  2145          CLRL    R0                              ; Assume not in "run" state
                 04     E1  0C3C  2146          BBC     #LPD$V_RUN,-                    ; If BC then no partner node
            3F 22 A6        0C3E  2147                      LPD$W_STS(R6),20$           ;
                 0A     E0  0C41  2148          BBS     #LPD$V_BC,-                     ; If a broadcast circuit,
            3A 22 A6        0C43  2149                      LPD$W_STS(R6),20$           ; then no block size
          57    20 A6     9A  0C46  2150          MOVZBL  LPD$B_PTH_INX(R6),R7            ; Get LPD index
    54  00000000'EF     D0  0C4A  2151          MOVL    NET$GL_PTR_VCB,R4               ; Get RCB address
        57    2C B447   D0  0C51  2152          MOVL    @RCB$L_PTR_ADJ(R4)[R7],R7       ; Get ADJ address
          58    04 A7   3C  0C56  2153          MOVZWL  ADJ$W_PNA(R7),R8                ; Get partner's node address
    5B  00000000'EF     D0  0C5A  2154          MOVL    NET$GL_CNR_NDI,R11              ; Get NDI root block
                F39C'   30  0C61  2155          BSBW    NET$NDI_BY_ADD                  ; Get the associated NDI CNF
              19 50     E9  0C64  2156          BLBC    R0,20$                          ; If LBC then none
                        0C67  2157          $GETFLD ndi,s,nna                       ; Get node name
                09 50   E9  0C74  2158          BLBC    R0,20$                          ; If LBC then not set
          63    68   57   28  0C77  2159          MOVC3   R7,(R8),(R3)                    ; Move the name
          50    0000'8F   3C  0C7B  2160          MOVZWL  #SS$_NORMAL,R0                  ; Indicate success
                   05   0C80  2161 20$:     RSB                                     ; Return status in R0
                        0C81  2162
                        0C81  2163 NET$CRI_S_LOO::                                  ; Get line's loopback nodename
                        0C81  2164          $GETFLD cri,s,nam                       ; Get line name
                34 50   E9  0C8E  2165          BLBC    R0,20$                          ; If LBC then error
      5B  00000000'EF   D0  0C91  2166          MOVL    NET$GL_CNR_NDI,R11              ; Get NDI root block
                   5A   D4  0C98  2167          CLRL    R10                             ; Indicate no current NDI CNF
                        0C9A  2168          $SEARCH eql,ndi,s,nli                   ; Get find NDI with this loopback line
                19 50   E9  0CA9  2169          BLBC    R0,20$                          ; If LBC then not found
                        0CAC  2170          $GETFLD ndi,s,nna                       ; Get the node name
                09 50   E9  0CB9  2171          BLBC    R0,20$                          ; If LBC then not set
          63    68   57   28  0CBC  2172          MOVC3   R7,(R8),(R3)                    ; Move the name
          50    0000'8F   3C  0CC0  2173          MOVZWL  #SS$_NORMAL,R0                  ; Indicate success
                   05   0CC5  2174 20$:     RSB                                     ; Return status in R0
                        0CC6  2175
                        0CC6  2176 NET$CRI_S_VMSNAM::                               ; Get VMS name of device
                        0CC6  2177          $GETFLD cri,s,nam                       ; Get network management circuit name
                03 50   E9  0CD3  2178          BLBC    R0,100$                         ; If LBC then error
                 0693   30  0CD6  2179          BSBW    TRAN_DEVNAM                     ; Get VMS device name
                        0CD9  2180                                                  ; Exit with status
                        0CD9  2181          ; Assume that any remaining characters are part of tributary identifier
                   05   0CD9  2182 100$:    RSB                                     ; Return status in R0
```

```
                        0CDA  2183
                        0CDA  2184 NET$CRI_S_DEVNAM::                   ; Get VMS device name, including unit
            F323'  30   0CDA  2185          BSBW    NET$LOCATE_LPD      ; Locate LPD
          59 50    E9   0CDD  2186          BLBC    R0,80$              ; Branch if none
             04    E1   0CE0  2187          BBC     #LPD$V_RUN,-        ; Exit if circuit not in run state
       54 22 A6         0CE2  2188                  LPD$W_STS(R6),80$
          5E 04    C2   0CE5  2189          SUBL    #4,SP               ; Used for returning device characteristics
          54 5E    D0   0CE8  2190          MOVL    SP,R4               ; Point R4 at return buffer
             00    DD   0CEB  2191          PUSHL   #0                  ; Terminate GETDVI item list
  00000029'EF    9F   0CED  2192          PUSHAB  QIOW_Q_IOSB         ; Point to DEVCHAR length word (use IOSB)
             54    DD   0CF3  2193          PUSHL   R4                  ; Point to DEVCHAR return buffer
          7E 02    B0   0CF5  2194          MOVW    #DVI$_DEVCHAR,-(SP) ; Item code
          7E 04    B0   0CF8  2195          MOVW    #4,-(SP)            ; Size of DEVCHAR result buffer
  00000031'EF    9F   0CFB  2196          PUSHAB  QIOW_W_RETLEN       ; Point to result length word
             53    DD   0D01  2197          PUSHL   R3                  ; Point to result buffer
          7E 20    B0   0D03  2198          MOVW    #DVI$_DEVNAM,-(SP)  ; Item code
          7E 20    B0   0D06  2199          MOVW    #DEVNAM_C_SIZ,-(SP) ; Size of result buffer
          51 5E    D0   0D09  2200          MOVL    SP,R1               ; Point to item list
                        0D0C  2201          $GETDVI_S ITMLST=(R1),-     ; Construct actual device name
                        0D0C  2202                    EFN=#NET$C_EFN_WAIT,- ; Use synchronous event flag
                        0D0C  2203                    CHAN=LPD$W_CHAN(R6)
          5E 20    C0   0D23  2204          ADDL    #8*4,SP             ; Pop item list
          15 50    E9   0D26  2205          BLBC    R0,90$              ; Branch if error detected
       0C 64 0D    E1   0D29  2206          BBC     #DEV$V_NET,(R4),80$ ; Br if not a network device
51  00000031'EF    3C   0D2D  2207          MOVZWL  QIOW_W_RETLEN,R1    ; Get length of result buffer
          53 51    C0   0D34  2208          ADDL    R1,R3               ; Point past result string
             05    11   0D37  2209          BRB     90$                 ; Success
    50  0000'8F    3C   0D39  2210 80$:     MOVZWL  #SS$_IVDEVNAM,R0    ; Error in <cri,s,nam> field, or
                        0D3E  2211                                      ;   not a network device
                     05 0D3E  2212 90$:     RSB                         ; Return status in R0
                        0D3F  2213
                        0D3F  2214 NET$CRI_S_CNT::                      ; Get line counters (maybe clear them)
             53    DD   0D3F  2215          PUSHL   R3                  ; Save starting address of counter block
                        0D41  2216          $GETFLD cri,l_typ           ; Get circuit type
          31 50    E9   0D4E  2217          BLBC    R0,20$              ; Branch if not set, assume not X25
       03    58    D1   0D51  2218          CMPL    R8,#NMA$C_CIRTY_X25 ; X.25 circuit?
          2C       12   0D54  2219          BNEQ    20$                 ; Branch if not
        F2A7'  30   0D56  2220          BSBW    NET$LOCATE_LPD          ; Get associated LPD, if any
          21 50    E9   0D59  2221          BLBC    R0,10$              ; Branch if no datalink mapping enabled
                        0D5C  2222          ;
                        0D5C  2223          ; X.25 datalink mapping circuit counters
                        0D5C  2224          ;
             3F    10   0D5C  2225          BSBB    50$                 ; Get Transport counters & seconds
             53    DD   0D5E  2226          PUSHL   R3                  ; Save current position in buffer
          0065     30   0D60  2227          BSBW    200$                ; Get PSI circuit counters & seconds
             58 8ED0    0D63  2228          POPL    R8                  ; Get start of PSI counter block
          0F 50    E9   0D66  2229          BLBC    R0,5$               ; Branch if no PSI counters (possible
                        0D69  2230                                      ; for switched circuits)
    C000 8F    68    B1   0D69  2231          CMPW    (R8),#NET$C_NMACNT_SLZ ; "seconds since last zeroed" ?
                1A    12   0D6E  2232          BNEQ    30$                 ; Branch if not
             57 04    C2   0D70  2233          SUBL    #4,R7               ; Account for loss of PSI "seconds"
68     04 A8 57    28   0D73  2234          MOVC    R7,4(R8),(R8)       ; Delete PSI seconds counter from block
             50 01    D0   0D78  2235 5$:      MOVL    #1,R0               ; Success
                0D    11   0D7B  2236          BRB     30$
                        0D7D  2237          ;
                        0D7D  2238          ; Native X.25 circuit counters
                        0D7D  2239          ;
```

```
                0048   30   0D7D  2240 10$:    BSBW    200$                    ; Get PSI circuit counters & seconds
                  08   11   0D80  2241         BRB     30$                     ; and continue with status in R0
                       0D82  2242         ;
                       0D82  2243         ; Non-X25 circuit counters
                       0D82  2244         ;
                F27B'  30   0D82  2245 20$:    BSBW    NET$LOCATE_LPD          ; Get associated LPD
              11 50    E9   0D85  2246         BLBC    R0,40$                  ; Error if LBC
                 13    10   0D88  2247         BSBB    50$                     ; Get the Transport counters
           52    6E    D0   0D8A  2248 30$:    MOVL    (SP),R2                 ; Recover original counter block ptr
             09 50    E9   0D8D  2249         BLBC    R0,40$                  ; If LBC then error
           50    03    D0   0D90  2250         MOVL    S^#EVC$C_SRC_CIR,R0     ; Setup database event i.d.
              F26A'  30   0D93  2251         BSBW    LOG_COUNTERS            ; Log the counters if they were zeroed
           50    00'   D0   0D96  2252         MOVL    S^#SS$_NORMAL,R0        ; Success
           5E    04    C0   0D99  2253 40$:    ADDL    #4,SP                   ; Pop buffer pointer
                 05    0D9C  2254         RSB                             ; Done
                       0D9D  2255         ;
                       0D9D  2256         ;
                       0D9D  2257         ;   Subroutine to move Transport and datalink counters
                       0D9D  2258         ;
        55   00000000'EF   9E   0D9D  2259 50$:    MOVAB   LPD_CNT_TAB,R5          ; Point to Transport counter table
                52    1A   3C   0DA4  2260         MOVZWL  #4+LPD$C_CNT_SIZE,R2    ; Number of bytes of Transport counters
                       0DA7  2261                                                 ; +4 for ABS_TIME
              51    36 A6   9E   0DA7  2262         MOVAB   LPD$L_ABS_TIM(R6),R1    ; Point to counters
            00000000'EF   16   0DAB  2263         JSB     MOVE_FMT_CNT            ; Move and format Transport counters
                       0DB1  2264         ;
                       0DB1  2265         ;   Get the counters from the driver and append to current counters
                       0DB1  2266         ;   minus the 'seconds since last zeroed' counter.
                       0DB1  2267         ;
           0E 22 A6    07   E0   0DB1  2268         BBS     #LPD$V_X25,LPD$W_STS(R6),90$ ; Skip if X.25 datalink
              52    14 A6   3C   0DB6  2269         MOVZWL  LPD$W_CHAN(R6),R2       ; Is there an I/O channel
                    08   13   0DBA  2270         BEQL    90$                     ; If EQL no, skip this phase
                 0000'8F   3C   0DBC  2271         MOVZWL  #IO$_SENSEMODE!-        ; Setup function code
                    50         0DC0  2272                   IO$M_RD_COUNT,R0      ;
                  0673   30   0DC1  2273         BSBW    DEV_CNT_QIO             ; Issue circuit QIO for counters
                 50    0C'   3C   0DC4  2274 90$:    MOVZWL  S^#SS$_NORMAL,R0        ; Indicate success
                       05   0DC7  2275 100$:   RSB                             ; Done
                       0DC8  2276         ;
                       0DC8  2277         ;
                       0DC8  2278         ; Subroutine to move PSI circuit counters
                       0DC8  2279         ;
                       0DC8  2280         ;
                 0048  8F   BB   0DC8  2281 200$:   PUSHR   #^M<R3,R6>              ; Save registers
                       0DCC  2282         $CNFFLD cri,s,nam,R9            ; Set field ID of circuit name
                   FB55   30   0DD3  2283         BSBW    SEND_TO_PSI             ; Call co-routine to setup NFB, etc.
                 22 50    E9   0DD6  2284         BLBC    R0,59$                  ; Branch if error detected
                       0DD9  2285         $CNFFLD cri,s,cnt,(R2)+         ; Store counter field ID into NFB
                 66    22   90   0DE0  2286         MOVB    #NFB$C_FC_SHOW,NFB$B_FCT(R6) ; Set function code
     03 00000000'EF    02   E1   0DE3  2287         BBC     #NET$V_CLRCNT,NET$GL_FLAGS,55$ ; Branch if not clearing counters
                 66    25   90   0DEB  2288         MOVB    #NFB$C_FC_ZERCOU,NFB$B_FCT(R6) ; Set READ & ZERO function code
        53    00000200  8F   C0   0DEE  2289 55$:    ADDL    #P4BUF_SIZE,R3          ; Set size of P4 return buffer
                    9E   16   0DF5  2290         JSB     @(SP)+                  ; Call SEND_TO_PSI back to issue QIO
                 0048  8F   BA   0DF7  2291         POPR    #^M<R3,R6>              ; Restore registers
                   C9  50   E9   0DFB  2292 59$:    BLBC    R0,100$                 ; Branch if error detected
                 57    88   3C   0DFE  2293         MOVZWL  (R8)+,R7                ; Make descriptor of CNT string
              63    68    57   28   0E01  2294         MOVC    R7,(R8),(R3)            ; Copy P4 buffer into counter block
                 50    59   D0   0E05  2295         MOVL    R9,R0                   ; Point to scratch storage
               00000000'EF   16   0E08  2296         JSB     NET$DEALLOCATE          ; Deallocate SHOW buffer
```

```
                50   01    D0  0E0E  2297          MOVL    #1,R0                   ; Success
                           05  0E11  2298          RSB
                               0E12  2299
                               0E12  2300 NET$CRI_S_CHR::                         ; Get line's startup characteristics
                     F1EB'     30  0E12  2301          BSBW    NET$LOCATE_LPD      ; Get associated LPD
                   45 50       E9  0E15  2302          BLBC    R0,100$             ; Error if LBC
                      53       DD  0E18  2303          PUSHL   R3                  ; Save buffer pointer
             00       6E       00  2C  0E1A  2304      MOVC5   #0,(SP),#0,-        ; Zero buffer header
                      63       18      0E1E  2305              #DLLQIO$C_LENGTH,(R3) ; ...advance R3
                         55 8ED0      0E20  2306      POPL    R5                  ; Point to DLLQIO block
                         50   D4      0E23  2307      CLRL    R0                  ;& (better code) Assume no PLVEC index
                54       28 A6   9A  0E25  2308      MOVZBL  LPD$B_PLVEC(R6),R4  ; Get PLVEC index
                         32      13  0E29  2309      BEQL    100$                ; Report error
                50       10 A6   D0  0E2B  2310      MOVL    LPD$L_UCB(R6),R0    ; Get device UCB
                         2C      13  0E2F  2311      BEQL    100$                ; If EQL then none
                    0000'8F      3C  0E31  2312      MOVZWL  #IO$_SETMODE,-      ; Setup I/O function code
                         65          0E35  2313                  D[LQIO$L_FUNC(R5)    ;
             52  0000016C'EF      9E  0E36  2314      MOVAB   CRI_TRN_TAB,R2      ; Setup parameter translation table
                      5D'AF      9F  0E3D  2315      PUSHAB  B^10$               ; Setup return address
                                 0E40  2316      $DISPATCH TYPE=B,PLVEC$AB_DEV[R4],- ; Dispatch on device type
                                 0E40  2317          <-
                                 0E40  2318              <DEVTRN$C_DEV_DMC,      CRI_DMC>,-
                                 0E40  2319              <DEVTRN$C_DEV_PCL,      CRI_PCL>,-
                                 0E40  2320              <DEVTRN$C_DEV_DMP,      CRI_DMP>,-
                                 0E40  2321              <DEVTRN$C_DEV_CI,       CRI_CI>,-
                                 0E40  2322              <DEVTRN$C_DEV_UNA,      CRI_UNA>,-
                                 0E40  2323          >
                         01      11  0E5B  2324      BRB     CRI_FOREIGN         ; Br if device is unknowm
                                 0E5D  2325 10$:
                               05  0E5D  2326 100$:   RSB
                                 0E5E  2327
                                 0E5E  2328 CRI_DMP:
                                 0E5E  2329 CRI_PCL:
                                 0E5E  2330 CRI_CI:
                                 0E5E  2331 CRI_UNA:
                                 0E5E  2332 CRI_FOREIGN:
                      044F       30  0E5E  2333          BSBW    BUILD_DEVBUF        ; Build the device parameter buffer
                               05  0E61  2334          RSB                         ;
                                 0E62  2335
                                 0E62  2336 CRI_DMC:
          04 A5       53   55   C3  0E62  2337          SUBL3   R5,R3,DLLQIO$L_P1(R5) ; Setup offset to characteristics buffer
                                 0E67  2338          ;
                                 0E67  2339          ;
                                 0E67  2340          ;   Get the characteristics buffer form the UCB.  This buffer was setup
                                 0E67  2341          ;   properly (with the exception of the MOP bit) from the PLI database
                                 0E67  2342          ;   when the corresponding DMC "line" was turned on.
                                 0E67  2343          ;
                                 0E67  2344          ;
                                 0E67  2345          ASSUME  UCB$L_DEVDEPEND  EQ  2+UCB$W_DEVBUFSIZ
                                 0E67  2346          ASSUME  UCB$W_DEVBUFSIZ  EQ  1+UCB$B_DEVTYPE
                                 0E67  2347          ASSUME  UCB$B_DEVTYPE    EQ  1+UCB$B_DEVCLASS
                                 0E67  2348
                83       40 A0   7D  0E67  2349          MOVQ    UCB$B_DEVCLASS(R0),(R3)+; Enter device characteristics
                                 0E6B  2350          CLRBIT  XM$V_CHR_MOP,-4(R3) ; Assume MOP mode not needed
                            02   E1  0E6F  2351          BBC     #LPD$V_DCE,-        ; If BC not in use for direct-access
                      04 22 A6       0E71  2352                      LPD$W_STS(R6),10$ ; functions
                                 0E74  2353          SETBIT  XM$V_CHR_MOP,-4(R3) ; Put line in MOP mode
```

```
                                                H 3
                     0E78  2354 10$:    ;
                     0E78  2355         ;
                     0E78  2356         ;   Get number of receive buffers from the PLI database
                     0E78  2357         ;
                     0E78  2358         ;
        0C02 8F   BB 0E78  2359         PUSHR   #^M<R1,R10,R11>         ; Save regs
5B 00000000'EF    D0 0E7C  2360         MOVL    NET$GL_CNR_PLI,R11      ; Get PLI root block
           5A     D4 0E83  2361         CLRL    R10                     ; Search from begining of list
        58 54     D0 0E85  2362         MOVL    R4,R8                   ; Search key is the PLVEC index
                     0E88  2363         $SEARCH eql,pli,l,plvec         ; Find PLI's CNF block
          10 50    E9 0E97  2364         BLBC    R0,15$                  ; If LBC then not found
                     0E9A  2365         $GETFLD pli,l,bfn               ; Get # of receive buffers
          03 50    E8 0EA7  2366         BLBS    R0,20$                  ; If LBS then field was active
        58 04     D0 0EAA  2367 15$:    MOVL    #4,R8                   ; Setup default
     0C A5 58     D0 0EAD  2368 20$:    MOVL    R8,DLLQIO$L_P3(R5)      ; Setup # of receive buffers
        0C02 8F   BA 0EB1  2369         POPR    #^M<R1,R10,R11>         ; Restore regs
                     0EB5  2370         ;
        50 01     90 0EB5  2371         MOVB    #1,R0                   ; Indicate success
                     05 0EB8  2372 100$:   RSB                           ; Return status in R0
```

```
                        .SBTTL  PLI parameter action routines
                ;+
                ; NET$PLI_V_LCK -  Get status of conditionally writeable fields
                ;
                ; NET$PLI_L_SUB -  Get line's substate
                ; NET$PLI_L_BUS -  Get line's receive buffer size
                ; NET$PLI_L_PLVEC  Get line's PLVEC index
                ;
                ; NET$PLI_S_COL -  Get line's collating value
                ; NET$PLI_S_VMSNAM Get VMS name of associated device
                ; NET$PLI_S_DEVNAM Get VMS name of associated device, but with unit included
                ; NET$PLI_S_CNT -  Get (optionally clear) line counters
                ; NET$PLI_S_CHR -  Get line's characteristics buffer
                ;
                ; INPUTS:       R11     PLI CNR address
                ;               R10     PLI CNF address
                ;               R9      FLD i.d. of field being read
                ;               R1      Scratch
                ;               R0      Scratch
                ;
                ; OUTPUTS:      R1      Address of string descriptor or address of field value
                ;               R0      Low bit set if R1 is valid
                ;                       Low bit clear otherwise
                ;
                ;               All other register values are preserved.
                ;
                ;-

                NET$PLI_V_LCK::                         ; Get status of cond. writeable fields
                        $GETFLD pli,l,sta               ; Get the state value
07 50   E9      BLBC    R0,10$                          ; If LBC then not locked
58 01   D1      CMPL    #NMA$C_STATE_OFF,R8             ; Is the line turned off?
   02   12      BNEQ    10$                             ; If NEQ then locked
   50   D4      CLRL    R0                              ; Say "not locked"
51 50   D0      MOVL    R0,R1                           ; Return as field value
50 00'  D0      MOVL    S^#SS$_NORMAL,R0                ; Success
        05      RSB

                NET$PLI_L_SUB::                         ; Get line's substate
        50  D4  CLRL    R0                              ; No substate is defined for lines
                                                        ; at this time
        05      RSB                                     ; Return status in R0

                NET$PLI_L_PLVEC::                       ; Get line's PLVEC index
51 12 AA  9A    MOVZBL  CNF$W_ID(R10),R1                ; Get the PLVEC index
50 00'  D0      MOVL    S^#SS$_NORMAL,R0                ; Success
        05      RSB                                     ; Return

                NET$PLI_L_BUS::                         ; Get line's receive buffer size
                                                        ; (MUST preserve R2-R11)
    0380 8F BB  PUSHR   #^M<R7,R8,R9>                   ; Save registers
51 00000000'EF  D0      MOVL    NET$GL_PTR_VCB,R1       ; Get RCB address
  51   7E A1    3C      MOVZWL  RCB$W_TOTBUFSIZ(R1),R1  ; Store the local block size
51 004C 8F      A2      SUBW    #CXB$C_OVERHEAD,R1      ; Subtract out overhead
                        ;
                        ;   If this is an Ethernet controller, then add an additional
                        ;   15 bytes overhead to account for the extended Ethernet
```

```
                        0EF6  2431          ;   Route Header used by Transport.  This is done only for
                        0EF6  2432          ;   Ethernet datalinks to avoid burdening point-to-point datalinks
                        0EF6  2433          ;   with the additional memory for its buffers.
                        0EF6  2434          ;
          50   12 AA    3C  0EF6  2435          MOVZWL  CNF$W_ID(R10),R0        ; Get PLVEC index
                  0D    13  0EFA  2436          BEQL    50$                     ; If none, skip it
09  00000000'EF40       91  0EFC  2437          CMPB    PLVEC$AB_DEV[R0],-      ; Is this an Ethernet controller?
                        0F04  2438                  #DEVTRN$C_DEV_UNA
                  03    12  0F04  2439          BNEQ    50$                     ; Skip if not
             51   16    C0  0F06  2440          ADDL    #TR$C_MAXHDR-6,R1       ; Add additional overhead for
                        0F09  2441                                              ; Phase IV Ethernet Transport header
                        0F09  2442          ;
                        0F09  2443          ;   If the user wants a larger buffer size for this line,
                        0F09  2444          ;   then return that value rather than the default executor
                        0F09  2445          ;   buffer size.
                        0F09  2446          ;
                        0F09  2447  50$:    $GETFLD pli,l,bfs               ; Executor buffer size overridden?
                03 50   E9  0F16  2448          BLBC    R0,90$                  ; Skip if not set
             51   58    D0  0F19  2449          MOVL    R8,R1                   ; If set, use it (includes xpt header)
          50   0000'8F   3C  0F1C  2450  90$:    MOVZWL  #SS$_NORMAL,R0          ; Success
               0380 8F   BA  0F21  2451          POPR    #^M<R7,R8,R9>           ; Restore registers
                        05  0F25  2452          RSB                             ; Return status in R0
                        0F26  2453
                        0F26  2454  NET$PLI_S_COL::                             ; Get line's collating value
                        0F26  2455          $GETFLD pli,s,nam               ; Get line's name
                09 50   E9  0F33  2456          BLBC    R0,10$                  ; Branch if not present
          63   68   57   28  0F36  2457          MOVC3   R7,(R8),(R3)            ; Move the name
          50   0000'8F   3C  0F3A  2458          MOVZWL  #SS$_NORMAL,R0          ; Indicate success
                        05  0F3F  2459  10$:    RSB                             ; Return status in R0
                        0F40  2460
                        0F40  2461  NET$PLI_S_VMSNAM::                          ; Get VMS name of device
                        0F40  2462          $GETFLD pli,s,nam               ; Get network management line name
                0F 50   E9  0F4D  2463          BLBC    R0,10$                  ; If LBC then error
                 0419   30  0F50  2464          BSBW    TRAN_DEVNAM             ; Translate the device name
                09 50   E9  0F53  2465          BLBC    R0,10$                  ; If LBC then error
                   57   D5  0F56  2466          TSTL    R7                      ; Any characters left?
                   05   13  0F58  2467          BEQL    10$                     ; If EQL then none, name is okay
          50   0000'8F   3C  0F5A  2468          MOVZWL  #SS$_IVDEVNAM,R0        ; Error in <pli,s,nam> field
                        05  0F5F  2469  10$:    RSB                             ; Return status in R0
                        0F60  2470
                        0F60  2471  NET$PLI_S_DEVNAM::                          ; Get VMS device name, including unit
             58   12 AA    3C  0F60  2472          MOVZWL  CNF$W_ID(R10),R8        ; Get PLVEC index
                   5B    13  0F64  2473          BEQL    80$                     ; Branch if none
50  00000000'EF48           3C  0F66  2474          MOVZWL  PLVEC$AW_CHAN[R8],R0    ; Get channel used by line
                5E   04    C2  0F6E  2475          SUBL    #4,SP                   ; Used for returning device characteristics
                54   5E    D0  0F71  2476          MOVL    SP,R4                   ; Point R4 at return buffer
                     00    DD  0F74  2477          PUSHL   #0                      ; Terminate GETDVI item list
            00000029'EF    9F  0F76  2478          PUSHAB  QIOW_Q_IOSB             ; Point to DEVCHAR length word (use IOSB)
                     54    DD  0F7C  2479          PUSHL   R4                      ; Point to DEVCHAR return buffer
                7E   02    B0  0F7E  2480          MOVW    #DVI$_DEVCHAR,-(SP)     ; Item code
                7E   04    B0  0F81  2481          MOVW    #4,-(SP)                ; Size of DEVCHAR result buffer
            00000031'EF    9F  0F84  2482          PUSHAB  QIOW_W_RETLEN           ; Point to result length word
                     53    DD  0F8A  2483          PUSHL   R3                      ; Point to result buffer
                7E   20    B0  0F8C  2484          MOVW    #DVI$_DEVNAM,-(SP)      ; Item code
                7E   20    B0  0F8F  2485          MOVW    #DEVNAM_C_SIZ,-(SP)     ; Size of result buffer
                51   5E    D0  0F92  2486          MOVL    SP,R1                   ; Point to item list
                        0F95  2487          $GETDVI_S ITMLST=(R1),-         ; Construct actual device name
```

```
                                0F95  2488                    EFN=#NET$C_EFN_WAIT,-   ; Use synchronous event flag
                                0F95  2489                    CHAN=R0
                     5E   20    C0  0FAB  2490            ADDL    #8*4,SP                 ; Pop item list
                       15 50    E9  0FAE  2491            BLBC    R0,90$                  ; Branch if error detected
                   0C 64   0D   E1  0FB1  2492            BBC     #DEV$V_NET,(R4),80$     ; Br if not a network device
               51  00000031'EF   3C  0FB5  2493            MOVZWL  QIOW_W_RETLEN,R1        ; Get length of result buffer
                     53   51    C0  0FBC  2494            ADDL    R1,R3                   ; Point past result string
                          05    11  0FBF  2495            BRB     90$                     ; Success
                50   0000'8F    3C  0FC1  2496 80$:       MOVZWL  #SS$_IVDEVNAM,R0        ; Error in <pli,s,nam> field or
                                0FC6  2497                                                ;   not a network device
                          05    05  0FC6  2498 90$:       RSB                             ; Return status in R0
                                0FC7  2499
                                0FC7  2500 NET$PLI_S_CNT::                                ; Get line counters (maybe clear them)
                          53    DD  0FC7  2501            PUSHL   R3                      ; Save starting address of counter block
                                0FC9  2502            ;
                                0FC9  2503            ;   If this is an X.25 circuit, then ask PSI for its counters.
                                0FC9  2504            ;
                                0FC9  2505            $GETFLD pli,l_pro               ; Get line protocol code
                       09 50    E9  0FD6  2506            BLBC    R0,10$                  ; Branch if not specified
                     05   58    D1  0FD9  2507            CMPL    R8,#NMA$C_LINPR_LAPB    ; LAPB line?
                          04    12  0FDC  2508            BNEQ    10$                     ; If not, use normal SENSEMODE
                          55    10  0FDE  2509            BSBB    200$                    ; Get PSI line counters & seconds
                          12    11  0FE0  2510            BRB     20$
                                0FE2  2511            ;
                                0FE2  2512            ; Get non-X25 line counters
                                0FE2  2513            ;
                  56    12 AA   3C  0FE2  2514 10$:       MOVZWL  CNF$W_ID(R10),R6        ; Get PLVEC index
                          18    13  0FE6  2515            BEQL    30$                     ; Branch if none
               01  00000000'EF46   91  0FE8  2516            CMPB    PLVEC$AB_DEV[R6],-      ; No counters for DMC "lines" (special
                                0FF0  2517                    #DEVTRN$C_DEV_DMC       ; case this since XMDRIVER does not
                          0E    13  0FF0  2518            BEQL    30$                     ; support the standard datalink driver
                                0FF2  2519                                            ; interface).
                          13    10  0FF2  2520            BSBB    50$                     ; Get the counters
                     52   6E    D0  0FF4  2521 20$:       MOVL    (SP),R2                 ; Recover original counter block ptr
                       09 50    E9  0FF7  2522            BLBC    R0,40$                  ; If LBC then error
                     50   01    D0  0FFA  2523            MOVL    S^#EVC$C_SRC_LIN,R0     ; Setup database event i.d.
                        F000'   30  0FFD  2524            BSBW    LOG_COUNTERS            ; Log the counters if they were zeroed
                     50   00'    D0  1000  2525 30$:       MOVL    S^#SS$_NORMAL,R0        ; Success
                     5E   04    C0  1003  2526 40$:       ADDL    #4,SP                   ; Pop counter address off stack
                          05    05  1006  2527            RSB                             ; Done
                                1007  2528
                                1007  2529 50$:       ;
                                1007  2530            ;   Subroutine to build the counter block
                                1007  2531            ;
               51  00000000'EF46   DE  1007  2532            MOVAL   PLVEC$AL_ABS_TIM[R6],R1 ; Point to seconds since last zeroed
                                100F  2533                                            ; counters
                     52   04    D0  100F  2534            MOVL    #4,R2                   ; Number of bytes in block
                 55  00000028'EF   9E  1012  2535            MOVAB   PLI_CNT_TAB,R5          ; Point to counter formatting table
                      00000000'EF   16  1019  2536            JSB     MOVE_FMT_CNT            ; Move and format the counters
                                101F  2537            ;
                                101F  2538            ;   Get the counters from the driver and append to counter block
                                101F  2539            ;   being constructed.
                                101F  2540            ;
               52  00000000'EF46   3C  101F  2541 70$:       MOVZWL  PLVEC$AW_CHAN[R6],R2    ; Is there an I/O channel
                          08    13  1027  2542            BEQL    90$                     ; If EQL no, skip this phase
                                3C  1029  2543            MOVZWL  #IO$_SENSEMODE!-        ; Setup function code
                                102A  2544                      IO$M_CTRL!-           ;
```

```
                50   0000'8F        102A  2545                    IO$M_RD_COUNT,R0        ;
                       0406   30     102E  2546            BSBW    DEV_CNT_QIO             ; Issue circuit QIO for counters
                     50   00'  3C     1031  2547 90$:       MOVZWL  S^#SS$_NORMAL,R0        ; Indicate success
                          05   05     1034  2548 100$:      RSB                             ; Done
                                      1035  2549
                                      1035  2550 ;
                                      1035  2551 ; Subroutine to get PSI LAPB line counters
                                      1035  2552 ;
                     0048 8F    BB    1035  2553 200$:      PUSHR   #^M<R3,R6>              ; Save registers
                                      1039  2554            $CNFFLD pli,s,nam,R9            ; Set field ID of line name
                       F8F8     30    1040  2555            BSBW    SEND_TO_PSI             ; Call co-routine to setup NFB, etc.
                        22 50   E9    1043  2556            BLBC    R0,59$                  ; Branch if error detected
                                      1046  2557            $CNFFLD pli,s,cnt,(R2)+         ; Store counter field ID into NFB
                     66   22    90    104D  2558            MOVB    #NFB$C_FC_SHOW,NFB$B_FCT(R6) ; Set function code
          03 00000000'EF   02    E1    1050  2559            BBC     #NET$V_CLRCNT,NET$GL_FLAGS,55$ ; Branch if not clearing counters
                     66   25    90    1058  2560            MOVB    #NFB$C_FC_ZERCOU,NFB$B_FCT(R6) ; Set READ & ZERO function code
                00000200 8F    C0    105B  2561 55$:       ADDL    #P4BUF_SIZE,R3          ; Set size of P4 return buffer
                          9E   16    1062  2562            JSB     @(SP)+                  ; Call SEND_TO_PSI back to issue QIO
                     0048 8F    BA    1064  2563            POPR    #^M<R3,R6>              ; Restore registers
                       C9 50    E9    1068  2564 59$:       BLBC    R0,100$                 ; Branch if error detected
                     57   88    3C    106B  2565            MOVZWL  (R8)+,R7                ; Make descriptor of CNT string
                63   68   57    28    106E  2566            MOVC    R7,(R8),(R3)            ; Copy string into counter block
                     50   59    D0    1072  2567            MOVL    R9,R0                   ; Point to scratch storage
                 00000000'EF    16    1075  2568            JSB     NET$DEALLOCATE          ; Deallocate SHOW buffer
                     50   01    D0    107B  2569            MOVL    #1,R0                   ; Success
                          05   05     107E  2570            RSB
                                      107F  2571
                                      107F  2572 NET$PLI_S_CHR::                            ; Get line's startup characteristics
                     53   DD    107F  2573            PUSHL   R3                      ; Save pointer to buffer
                00   6E   00    2C    1081  2574            MOVC5   #0,(SP),#0,-            ; Zero buffer header
                     63   18          1085  2575                    #DLLQIO$C_LENGTH,(R3)   ; ...advance R3
                     55 8ED0    1087  2576            POPL    R5                      ; Point to top of buffer
                     50   D4    108A  2577            CLRL    R0                      ; Assume no PLVEC index assigned
                  54   12 AA   3C    108C  2578            MOVZWL  CNF$W_ID(R10),R4        ; Get PLVEC index
                     4C   13    1090  2579            BEQL    100$                    ; If EQL then return error
                                      1092  2580            ;
                                      1092  2581            ;   Setup default function value
                                      1092  2582            ;
                                      1092  2583            $DISPATCH TYPE=B,PLVEC$AB_STATE[R4],-   ; Dispatch by state value
                                      1092  2584                <-
                                      1092  2585                <NMA$C_STATE_OFF,   10$>,-
                                      1092  2586                <NMA$C_STATE_ON,    20$>,-
                                      1092  2587                >
                50   0000'8F   B0    109F  2588            MOVW    #IO$_SETMODE!IO$M_CTRL,R0 ; Use this for all other states
                          0C   11    10A4  2589            BRB     30$                     ; Continue
                     0000'8F   B0    10A6  2590 10$:       MOVW    #IO$_SETMODE!IO$M_CTRL!-; State is "off"
                          50         10AA  2591                    IO$M_SHUTDOWN,R0        ;
                          05   11    10AB  2592            BRB     30$                     ; Continue
                     0000'8F   B0    10AD  2593 20$:       MOVW    #IO$_SETMODE!IO$M_CTRL!-; State is "on"
                          50         10B1  2594                    IO$M_STARTUP,R0         ;
                     65   50   3C    10B2  2595 30$:       MOVZWL  R0,DLLQIO$L_FUNC(R5)    ; Setup I/O function code
           52   000001B8'EF   9E    10B5  2596            MOVAB   PLI_TRN_TAB,R2          ; Point to param translation table
                     DE'AF   9F    10BC  2597            PUSHAB  B^40$                   ; Setup return address
                                      10BF  2598
                                      10BF  2599            $DISPATCH TYPE=B,PLVEC$AB_DEV[R4],- ; Dispatch on device type
                                      10BF  2600                <-
                                      10BF  2601                <DEVTRN$C_DEV_DMC,     PLI_DMC>,-
```

```
                     10BF  2602                 <DEVTRN$C_DEV_PCL,     PLI_PCL>,-
                     10BF  2603                 <DEVTRN$C_DEV_DMP,     PLI_DMP>,-
                     10BF  2604                 <DEVTRN$C_DEV_CI,      PLI_CI>,-
                     10BF  2605                 <DEVTRN$C_DEV_UNA,     PLI_UNA>,-
                     10BF  2606                 <DEVTRN$C_DEV_PPUNA,   PLI_PPUNA>,-
                     10BF  2607             >
           01   11   10DC  2608             BRB     PLI_FOREIGN                ; Br if device is unknown
                     10DE  2609 40$:
                05   10DE  2610 100$:    RSB
                     10DF  2611
                     10DF  2612 PLI_DMP:
                     10DF  2613 PLI_PCL:
                     10DF  2614 PLI_CI:
                     10DF  2615 PLI_FOREIGN:
           01CE      30   10DF  2616             BSBW    BUILD_DEVBUF               ; Build the parameter buffer
                     05   10E2  2617             RSB
                     10E3  2618
                     10E3  2619 PLI_UNA:
           01CA      30   10E3  2620             BSBW    BUILD_DEVBUF               ; Build standard parameter buffer
                53   DD   10E6  2621             PUSHL   R3                         ; Save position in P2 buffer
     83    0B18 8F   B0   10E8  2622             MOVW    #NMA$C_PCLI_PRM,(R3)+      ; Promiscuous mode = OFF
           83   01   D0   10ED  2623             MOVL    #NMA$C_STATE_OFF,(R3)+
     83    0B19 8F   B0   10F0  2624             MOVW    #NMA$C_PCLI_MLT,(R3)+      ; Multicast address state = OFF
           83   01   D0   10F5  2625             MOVL    #NMA$C_STATE_OFF,(R3)+
     83    0B1A 8F   B0   10F8  2626             MOVW    #NMA$C_PCLI_PAD,(R3)+      ; Padding on transmits = ON
           83   00   D0   10FD  2627             MOVL    #NMA$C_STATE_ON,(R3)+
     83    0B1B 8F   B0   1100  2628             MOVW    #NMA$C_PCLI_DCH,(R3)+      ; Data chaining = OFF
           83   01   D0   1105  2629             MOVL    #NMA$C_STATE_OFF,(R3)+     ; (DLLTRN can't handle multiple CXBs)
     83    0B1C 8F   B0   1108  2630             MOVW    #NMA$C_PCLI_CRC,(R3)+      ; CRC generation = ON
           83   00   D0   110D  2631             MOVL    #NMA$C_STATE_ON,(R3)+
     83    0B1E 8F   B0   1110  2632             MOVW    #NMA$C_PCLI_ACC,(R3)+      ; Protocol access mode = exclusive
           83   03   D0   1115  2633             MOVL    #NMA$C_ACC_EXC,(R3)+
     83    0B0E 8F   B0   1118  2634             MOVW    #NMA$C_PCLI_PTY,(R3)+      ; Protocol type parameter code
                      111D  2635             $GETFLD pli,l,ept                  ; Get protocol type to be used
           05   50   E8   112A  2636             BLBS    R0,10$                     ; Branch if ok
     58    0360 8F   3C   112D  2637             MOVZWL  #TR$C_NI_PROT,R8           ; Else, provide a default value
           83   58   D0   1132  2638 10$:       MOVL    R8,(R3)+                   ; Set protocol type
     83    0B04 8F   B0   1135  2639             MOVW    #NMA$C_PCLI_PHA,(R3)+      ; Physical address = Phase IV address
           83   08   B0   113A  2640             MOVW    #8,(R3)+                   ;    8 byte string:
           83   01   B0   113D  2641             MOVW    #NMA$C_LINMC_SET,(R3)+     ;    Set the address
83   000400AA 8F   D0   1140  2642             MOVL    #TR$C_NI_PREFIX,(R3)+      ;    Phase IV NI prefix
57   00000000'EF   D0   1147  2643             MOVL    NET$GL_PTR_VCB,R7          ;
     83   0E   A7   B0   114E  2644             MOVW    RCB$W_ADDR(R7),(R3)+       ;    Phase IV node number
     83    0B0F 8F   B0   1152  2645             MOVW    #NMA$C_PCLI_MCA,(R3)+      ; Multicast addresses are:
           83   08   B0   1157  2646             MOVW    #8,(R3)+                   ;    8 byte string:
           83   01   B0   115A  2647             MOVW    #NMA$C_LINMC_SET,(R3)+     ;    Set the address
83    030000AB 8F   D0   115D  2648             MOVL    #TR$C_NI_ALLROU1,(R3)+     ;    Phase IV "all routers" multicast
           83   00   B0   1164  2649             MOVW    #TR$C_NI_ALLROU2,(R3)+
     05    008A C7   91   1167  2650             CMPB    RCB$B_ETY(R7),#ADJ$C_PTY_PH4N ; If Phase IV endnode,
                0D   12   116C  2651             BNEQ    20$                        ;       use "all endnodes" instead
           53   06   C2   116E  2652             SUBL    #6,R3                      ;       of "all routers"
83    040000AB 8F   D0   1171  2653             MOVL    #TR$C_NI_ALLEND1,(R3)+     ;    Phase IV "all endnodes" multicast
           83   00   B0   1178  2654             MOVW    #TR$C_NI_ALLEND2,(R3)+
     52    53   8E   C3   117B  2655 20$:       SUBL3   (SP)+,R3,R2                ; Compute size of additional params
51     08 A5   55   C1   117F  2656             ADDL3   R5,DLLQIO$L_P2(R5),R1      ; Point to P2 buffer descriptor
           61   52   C0   1184  2657             ADDL    R2,(R1)                    ; Append these params to P2 buffer
           50   01   D0   1187  2658             MOVL    #1,R0                      ; Success
```

```
                                                     N 3
                     05  118A  2659          RSB
                         118B  2660
                         118B  2661 PLI_PPUNA_TAB:                    ; Parameter table for PPUNA pseudo-datalink
                         118B  2662          .cnvtab pcli,pli,BUS     ; Receive buffer size
                         118F  2663          .cnvtab pcli,pli,CON     ; Controller (loopback) mode
                         1193  2664          .cnvtab pcli,pli,BFN     ; Number of buffers in pool
               00000000  1197  2665          .LONG   0                ; Terminate the table
                         119B  2666
                         119B  2667 PLI_PPUNA:                                ; (used only for internal testing)
          52   ED AF    9E  119B  2668          MOVAB   PLI_PPUNA_TAB,R2      ; Point to param translation table
                 010E    30  119F  2669          BSBW    BUILD_DEVBUF          ; Build standard parameter buffer
                   53    DD  11A2  2670          PUSHL   R3                    ; Save position in P2 buffer
        83   0458 8F    B0  11A4  2671          MOVW    #NMA$C_PCLI_PRO,(R3)+ ; Protocol type = POINT
             83   00    D0  11A9  2672          MOVL    #NMA$C_LINPR_POI,(R3)+
        83   0B18 8F    B0  11AC  2673          MOVW    #NMA$C_PCLI_PRM,(R3)+ ; Promiscuous mode = OFF
             83   01    D0  11B1  2674          MOVL    #NMA$C_STATE_OFF,(R3)+
        83   0B19 8F    B0  11B4  2675          MOVW    #NMA$C_PCLI_MLT,(R3)+ ; Multicast address state = OFF
             83   01    D0  11B9  2676          MOVL    #NMA$C_STATE_OFF,(R3)+
        83   0B1A 8F    B0  11BC  2677          MOVW    #NMA$C_PCLI_PAD,(R3)+ ; Padding on transmits = ON
             83   00    D0  11C1  2678          MOVL    #NMA$C_STATE_ON,(R3)+
        83   0B1B 8F    B0  11C4  2679          MOVW    #NMA$C_PCLI_DCH,(R3)+ ; Data chaining = OFF
             83   01    D0  11C9  2680          MOVL    #NMA$C_STATE_OFF,(R3)+ ; (DLLTRN can't handle multiple CXBs)
        83   0B1C 8F    B0  11CC  2681          MOVW    #NMA$C_PCLI_CRC,(R3)+ ; CRC generation = ON
             83   00    D0  11D1  2682          MOVL    #NMA$C_STATE_ON,(R3)+
        83   0B1E 8F    B0  11D4  2683          MOVW    #NMA$C_PCLI_ACC,(R3)+ ; Protocol access mode = limited
             83   02    D0  11D9  2684          MOVL    #NMA$C_ACC_LIM,(R3)+
        83   0B21 8F    B0  11DC  2685          MOVW    #NMA$C_PCLI_DES,(R3)+ ; Remote destination = (dead timer)
             83   08    B0  11E1  2686          MOVW    #8,(R3)+              ;    8 byte string:
             83   01    B0  11E4  2687          MOVW    #NMA$C_LINMC_SET,(R3)+ ;    Set the address
  83   000400AA 8F    D0  11E7  2688          MOVL    #TR$C_NI_PREFIX,(R3)+ ;    Phase IV NI prefix
                         11EE  2689          $GETFLD pli,l,ddt             ; Get value of dead timer
                   0A    ED  11FB  2690          CMPZV   #TR4$V_ADDR_AREA,-    ; Is area portion of address 0?
       00   58    06        11FD  2691                  #TR4$S_ADDR_AREA,R8,#0
                   0E    12  1200  2692          BNEQ    5$                    ; If not, set it as is
  57   00000000'EF    D0  1202  2693          MOVL    NET$GL_PTR_VCB,R7     ;
             008B  C7    F0  1209  2694          INSV    RCB$B_HOMEAREA(R7),-  ; Else, insert our own area
                   0A        120D  2695                  #TR4$V_ADDR_AREA,-
             58    06        120E  2696                  #TR4$S_ADDR_AREA,R8   ; (leave R8 set for code below)
             83    58    B0  1210  2697 5$:      MOVW    R8,(R3)+              ;    Set remote destination address
        83   0B0E 8F    B0  1213  2698          MOVW    #NMA$C_PCLI_PTY,(R3)+ ; Protocol type parameter code
                         1218  2699          $GETFLD pli,l,ept             ; Get protocol type to be used
                 05 50   E8  1225  2700          BLBS    R0,10$                ; Branch if ok
        58   A002 8F    3C  1228  2701          MOVZWL  #^XA002,R8            ; Else, provide a default value
             83   58    D0  122D  2702 10$:     MOVL    R8,(R3)+              ; Set protocol type
        83   0B04 8F    B0  1230  2703          MOVW    #NMA$C_PCLI_PHA,(R3)+ ; Physical address = Phase IV address
             83   08    B0  1235  2704          MOVW    #8,(R3)+              ;    8 byte string:
             83   01    B0  1238  2705          MOVW    #NMA$C_LINMC_SET,(R3)+ ;    Set the address
  83   000400AA 8F    D0  123B  2706          MOVL    #TR$C_NI_PREFIX,(R3)+ ;    Phase IV NI prefix
  57   00000000'EF    D0  1242  2707          MOVL    NET$GL_PTR_VCB,R7     ;
         83   0E    A7  B0  1249  2708          MOVW    RCB$W_ADDR(R7),(R3)+  ;    Phase IV node number
       52   53      8E  C3  124D  2709          SUBL3   (SP)+,R3,R2           ; Compute size of additional params
     51   08 A5      55  C1  1251  2710          ADDL3   R5,DLLQIO$L_P2(R5),R1 ; Point to P2 buffer descriptor
             61      52  C0  1256  2711          ADDL    R2,(R1)               ; Append these params to P2 buffer
             50      01  D0  1259  2712          MOVL    #1,R0                 ; Success
                     05  125C  2713          RSB
                         125D  2714
                         125D  2715 PLI_DMC:
```

```
                                125D  2716          .IF NDF DMC_MODEM                ; Disallow controller functions if
                                125D  2717                                           ;  there is no modem support in DMC
                                125D  2718
                          51    D4  125D  2719          CLRL    R1                   ; Say "start with LPD index 0"
                        ED9E'   30  125F  2720          BSBW    NET$GET_PLVECLPD     ; Find next LPD actively using this line
                     05 50      E8  1262  2721          BLBS    R0,10$               ; If LBS then found one
                0000'8F         AA  1265  2722          BICW    #IO$M_SHUTDOWN,-     ; Can't issue "SHUTDOWN" if device is
                      65            1269  2723                  DLLQIO$L_FUNC(R5)    ; not active
                                AA  126A  2724 10$:     BICW    #IO$M_STARTUP!-      ; Special case -- DMC does not support
                                126B  2725                       IO$M_CTRL,-     ; these modifiers in for its "line"
              65  0000'8F       126B  2726                  DLLQIO$L_FUNC(R5)    ; aspect
                                126F  2727
                                126F  2728          .ENDC   ; DF DMC_MODEM
                                126F  2729
           04 A5  53    55    C3  126F  2730          SUBL3   R5,R3,DLLQIO$L_P1(R5) ; Setup offset to characteristics buffer
                  83    20    90  1274  2731          MOVB    #DC$_SCOM,(R3)+      ; Enter device class
                  83    01    90  1277  2732          MOVB    #DT$_DMC11,(R3)+     ; Enter device type
                                127A  2733          ;
                                127A  2734          ;   For now, all lines use the same buffer size.
                                127A  2735          ;
                        FC65    30  127A  2736          BSBW    NET$PLI_L_BUS        ; Get datalink buffer size
                  83      51    B0  127D  2737          MOVW    R1,(R3)+             ; Store the block size
                          63    D4  1280  2738          CLRL    (R3)                 ; Zero the characteristis
                                1282  2739          $GETFLD pli,v,con            ; Get loopback field value
                     05 58      E9  128F  2740          BLBC    R8,20$               ; If LBC not in controller loopback
                                1292  2741          SETBIT  XM$V_CHR_LOOPB,(R3)  ; Else set loopback bit
                          13    11  1295  2742          BRB     30$                  ; Don't bother with HDPLX
                                1297  2743 20$:     $GETFLD pli,v,dup            ; Get duplex selector
                     03 58      E9  12A4  2744          BLBC    R8,30$               ; LBS in R0 if half-duplex
                                12A7  2745          SETBIT  XM$V_CHR_HDPLX,(R3)  ; Put line in half-duplex
                          83    D5  12AA  2746 30$:     TSTL    (R3)+                ; Advance past characteristics
                  50      01    90  12AC  2747          MOVB    #1,R0                ; Indicate success
                                05  12AF  2748          RSB                          ; Return status in R0
```

```
                        12B0  2750          .SBTTL  BUILD_DEVBUF - Build DLLQIO buffer
                        12B0  2751 ;+
                        12B0  2752 ; BUILD_DEVBUF - Build P2 buffer for SETMODE of line or circuit
                        12B0  2753 ;
                        12B0  2754 ; This routine is called to setup a standard P2 buffer for a line or
                        12B0  2755 ; circuit SETMODE function.  The P2 buffer contains all the datalink
                        12B0  2756 ; parameters for the line/circuit to be initialized.
                        12B0  2757 ;
                        12B0  2758 ; Inputs:
                        12B0  2759 ;
                        12B0  2760 ;       R11 = PLI CNR address
                        12B0  2761 ;       R10 = PLI CNF address
                        12B0  2762 ;       R5 = Address of DLLQIO buffer
                        12B0  2763 ;       R3 = Address of scratch buffer
                        12B0  2764 ;       R2 = Address of datalink parameter translation table (CNVTAB)
                        12B0  2765 ;
                        12B0  2766 ; Outputs:
                        12B0  2767 ;
                        12B0  2768 ;       R3 = Updated pointer to end of scratch buffer
                        12B0  2769 ;       R0 = Status code
                        12B0  2770 ;-
                        12B0  2771
                        12B0  2772 BUILD_DEVBUF:                                ; Build device param buffer
    08 A5   53   D0     12B0  2773          MOVL    R3,DLLQIO$L_P2(R5)          ; Setup pointer to device parameter
                        12B4  2774                                              ; buffer descriptor
       53   08   C0     12B4  2775          ADDL    #8,R3                       ; Allow 8 bytes for the descriptor
    08 B5   53   CE     12B7  2776          MNEGL   R3,@DLLQIO$L_P2(R5)         ; Prepare for size calculation
FC A3   53   55   C3     12BB  2777          SUBL3   R5,R3,-4(R3)                ; Setup offset to param buffer
       0070 8F   BB     12C0  2778          PUSHR   #^M<R4,R5,R6>               ; Save regs
       56   52   D0     12C4  2779          MOVL    R2,R6                       ; Point to translation table
                        12C7  2780
            66   D5     12C7  2781 10$:     TSTL    (R6)                        ; At end of table?
            45   13     12C9  2782          BEQL    40$                         ; If EQL then yes
  3D 66     0C   E0     12CB  2783          BBS     #CNVTAB$V_RO,(R6),30$       ; Skip if should not send to driver
  7E   0A   AB   90     12CF  2784          MOVB    CNR$B_TYPE(R11),-(SP)       ; Enter database i.d.
            1E   EF     12D3  2785          EXTZV   #CNVTAB$V_FMT,-             ; Get param format type
51  66      02          12D5  2786                  #CNVTAB$S_FMT,(R6),R1       ;
    7E      51   90     12D8  2787          MOVB    R1,-(SP)                    ; Append to param database i.d.
            10   EF     12DB  2788          EXTZV   #CNVTAB$V_INT,-             ; Get internal param i.d. index
51  66      0E          12DD  2789                  #CNVTAB$S_INT,(R6),R1       ;
    7E      51   B0     12E0  2790          MOVW    R1,-(SP)                    ; Append param i.d.
            59 8ED0     12E3  2791          POPL    R9                          ; Get compelete param i.d., fix stack
3FFF 8F     59   B1     12E6  2792          CMPW    R9,#CNVTAB$C_INTRNL         ; Datalink only parameter?
            1F   13     12EB  2793          BEQL    30$                         ; If so, just ignore it
          ED10'  30     12ED  2794          BSBW    CNF$GET_FIELD               ; Get the field value
       19 50   E9     12F0  2795          BLBC    R0,30$                      ; If LBC then param is not set
            00   EF     12F3  2796          EXTZV   #CNVTAB$V_NMA,-             ; Get the NICE parameter code
50  66      0C          12F5  2797                  #CNVTAB$S_NMA,(R6),R0       ;
    83      50   B0     12F8  2798          MOVW    R0,(R3)+                    ; Enter the NICE param i.d.
    02      51   D1     12FB  2799          CMPL    R1,#NFB$C_TYP_STR           ; String ?
            05   13     12FE  2800          BEQL    20$                         ; If EQL yes
    83      58   D0     1300  2801          MOVL    R8,(R3)+                    ; Enter the parameter value
            07   11     1303  2802          BRB     30$                         ; Continue
    83      57   B0     1305  2803 20$:     MOVW    R7,(R3)+                    ; Enter the string length
63  68      57   28     1308  2804          MOVC3   R7,(R8),(R3)                ; Enter the string text, update R3
            86   D5     130C  2805 30$:     TSTL    (R6)+                       ; Advance to next table entry
            B7   11     130E  2806          BRB     10$                         ; Loop
```

```
                0070 8F    BA  1310  2807 40$:    POPR    #^M<R4,R5,R6>           ; Restore regs
            08 B5    53    C0  1314  2808         ADDL    R3,@DLLQIO$L_P2(R5)     ; Complete size calculation
            08 A5    55    C2  1318  2809         SUBL    R5,DLLQIO$L_P2(R5)      ; Convert pointer to offset
     50  00000000'8F       D0  131C  2810         MOVL    #SS$_NORMAL,R0          ; Setup status
                           05  1323  2811         RSB
```

```
                                1324  2813          .SBTTL  NET$CVT_NMA_INT - Convert NMA to NFB code
                                1324  2814 ;+
                                1324  2815 ; NET$CVT_NMA_INT - Convert an NMA code returned by a datalink driver
                                1324  2816 ; to a internal NFB parameter code.
                                1324  2817 ;
                                1324  2818 ; Inputs:        R11 = CNR address
                                1324  2819 ;                R9 = NMA code
                                1324  2820 ;
                                1324  2821 ; Outputs:       R9 = NFB code corresponding to NMA code
                                1324  2822 ;                     (or original, if could not map the code)
                                1324  2823 ;-
                                1324  2824 NET$CVT_NMA_INT::                           ; Convert NMA i.d. to internal i.d.
                          26    BB  1324  2825          PUSHR   #^M<R1,R2,R5>           ; Save regs
                55  0000016C'EF   9E  1326  2826          MOVAB   CRI_TRN_TAB,R5          ; Assume CRI database
                          04    91  132D  2827          CMPB    #NFB$C_DB_CRI,-         ; Circuit database ?
                        0A AB       132F  2828                  CNR$B_TYPE(R11)         ;
                          07    13  1331  2829          BEQL    10$                     ; If EQL then yes
                55  000001B8'EF   9E  1333  2830          MOVAB   PLI_TRN_TAB,R5          ; Else PLI database
                          03    10  133A  2831 10$:      BSBB    CVT_NMA_INT             ; Convert the i.d.
                          26    BA  133C  2832          POPR    #^M<R1,R2,R5>           ; Restore regs
                          05    13E  2833          RSB
                                133F  2834
                                133F  2835 ;
                                133F  2836 ; Do the actual conversion of NMA to NFB parameter codes.
                                133F  2837 ;
                                133F  2838 ; Inputs:
                                133F  2839 ;        R9 = NMA i.d. code, with format type
                                133F  2840 ;        R5 = Conversion table address
                                133F  2841 ;
                                133F  2842 ; Outputs:
                                133F  2843 ;        R1 = Last table entry used (zero if end of table)
                                133F  2844 ;        R0 = Status indicator
                                133F  2845 ;
                                133F  2846 ;        R2 is destroyed.
                                133F  2847 ;
                                133F  2848
                                133F  2849 CVT_NMA_INT:                             ; Convert NMA i.d. to internal i.d.
      52    59    0C    00    EF  133F  2850          EXTZV   #0,#CNVTAB$S_NMA,R9,R2  ; Get only NMA param i.d.
                          50    D4  1344  2851          CLRL    R0                      ; Assume conversion error
                      51    85    D0  1346  2852 40$:     MOVL    (R5)+,R1                ; Get next table entry
                          20    13  1349  2853          BEQL    100$                    ; If EQL table is exhausted, report
                                134B  2854                                              ; that the driver is not supported
                                134B  2855          ASSUME  NFB$V_INX       EQ   0
                                134B  2856          ASSUME  NFB$V_TYP       EQ  16
                                134B  2857          ASSUME  NFB$V_DB        EQ  24
                                134B  2858
                          00    ED  134B  2859          CMPZV   #CNVTAB$V_NMA,-
            52    51    0C          134D  2860                  #CNVTAB$S_NMA,R1,R2     ; Is this it?
                          F4    12  1350  2861          BNEQ    40$                     ; If NEQ then keep trying
                          10    EF  1352  2862          EXTZV   #CNVTAB$V_INT,-         ;
            7E    51    0E          1354  2863                  #CNVTAB$S_INT,R1,-(SP)  ; Put internal code on stack longword
                          1E    EF  1357  2864          EXTZV   #CNVTAB$V_FMT,-         ; Get field format type
            52    51    02          1359  2865                  #CNVTAB$S_FMT,R1,R2     ;
         02 AE    52    90  135C  2866          MOVB    R2,2(SP)                ; Insert format into TYP field
      03 AE    0A AB    90  1360  2867          MOVB    CNR$B_TYPE(R11),3(SP)   ; Insert database id into DB field
                      59  8ED0  1365  2868          POPL    R9                      ; Get full field i.d., cleanup stack
                      50    01    D0  1368  2869          MOVL    #1,R0                   ; Indicate successful conversion
```

```
05  136B  2870 100$:    RSB
```

```
                        136C  2872          .SBTTL  TRAN_DEVNAM - Translate device name
                        136C  2873 ;+
                        136C  2874 ; TRAN_DEVNAM   Translate Network Managment name to VMS device name
                        136C  2875 ;
                        136C  2876 ; This routine translates a network management line/circuit name to a VMS
                        136C  2877 ; device name.  Network management line names have the format "dev-c-u" and
                        136C  2878 ; circuit names have the format "dev-c-u.t"  where c,u,t are ascii-decimal
                        136C  2879 ; strings and "dev" is a network managment device numeric.
                        136C  2880 ;
                        136C  2881 ; INPUTS:        R8      Pointer to network management name
                        136C  2882 ;                R7      Number of characters in name
                        136C  2883 ;                R3      Output buffer pointer
                        136C  2884 ;
                        136C  2885 ; OUPUTS:        R8      Pointer to first unparsed character
                        136C  2886 ;                R7      Number of unparsed characters
                        136C  2887 ;                R6      Pointer to device table entry
                        136C  2888 ;                R5,R4   Garbage
                        136C  2889 ;                R3      Advanced by characters in translated name
                        136C  2890 ;                R2,R1   Garbage
                        136C  2891 ;                R0      Status
                        136C  2892 ;
                        136C  2893 ;                All other regs are perserved
                        136C  2894 ;-
                        136C  2895 TRAN_DEVNAM:
                        136C  2896          ;
                        136C  2897          ;   Parse the device mnemonic, locate translation table entry
                        136C  2898          ;
          4F   10       136C  2899          BSBB    PRS_MNEMONIC                ; Parse name, locate entry in table
     4B 50   E9         136E  2900          BLBC    R0,T10$                     ; If LBC then error
63   61 52   28         1371  2901          MOVC3   R2,(R1),(R3)                ; Move the VMS device mnemonic
  08   0A A6   91       1375  2902          CMPB    DEVTRN$B_DEV(R6),#DEVTRN$C_DEV_X25 ; X25 circuit name?
       33   13         1379  2903          BEQL    40$                         ; If so, then use only device name
                        137B  2904          ;
                        137B  2905          ;                                       dev-c-u.t
                        137B  2906          ;   We've parsed up to here  ^      Translate the controller specifier.
                        137B  2907          ;
       54   2E   9A     137B  2908          MOVZBL  #^A".",R4                   ; Setup field delimiter for single-unit
                        137E  2909                                              ; device (i.e., no unit field)
            00   E1     137E  2910          BBC     #DEVTRN$V_MULTI,-           ; If BC then not multi-unit device
     03 0C A6           1380  2911                    DEVTRN$B_CHAR(R6),20$      ; and there's no unit field
       54   2D   9A     1383  2912          MOVZBL  #^A"-",R4                   ; Setup field delimiter
            7E   10     1386  2913 20$:     BSBB    PRS_DECIMAL                 ; Convert the controller to binary
         31 50   E9     1388  2914          BLBC    R0,T10$                     ; If LBC then error
            52   D5     138B  2915          TSTL    R2                          ; Any decimal characters parsed ?
            28   13     138D  2916          BEQL    100$                        ; If EQL no, illegal name
       19   51   91     138F  2917          CMPB    R1,#25                      ; Within range (0=A, 25=Z)
            23   1A     1392  2918          BGTRU   100$                        ; If GTRU then out of range
83   51   41 8F   81     1394  2919          ADDB3   #^A'A',R1,(R3)+             ; Store as VMS controller designator
                        1399  2920          ;
                        1399  2921          ;                                   dev-c-u.t
                        1399  2922          ;   We have parsed up to here               ^      Parse the unit number.
                        1399  2923          ;
            50   D4     1399  2924          CLRL    R0                          ; Assume single-unit device
            00   E1     139B  2925          BBC     #DEVTRN$V_MULTI,-           ; If BC then single-unit device and
     0B 0C A6           139D  2926                    DEVTRN$B_CHAR(R6),30$      ; there's no unit field
       54   2E   9A     13A0  2927          MOVZBL  #^A".",R4                   ; Setup field delimiter
            61   10     13A3  2928          BSBB    PRS_DECIMAL                 ; Convert the Ascii unit to binary
```

```
                14 50   E9  13A5  2929            BLBC    R0,110$                 ; If LBC then illegal name
             50   51   D0  13A8  2930            MOVL    R1,R0                   ; Get the binary value
               EC52'   30  13AB  2931 30$:       BSBW    NET$BIN2ASC             ; Move to @R3 as Ascii
     50  00000000'8F   D0  13AE  2932 40$:       MOVL    #SS$_NORMAL,R0          ; Indicate success
                 05   11  13B5  2933             BRB     110$                    ; Take common exit
                          13B7  2934
         50   0000'8F   3C  13B7  2935 100$:     MOVZWL  #SS$_IVDEVNAM,R0        ; Error in translated name
                      05  13BC  2936 110$:      RSB                             ; Return status in R0
```

```
                        13BD  2938          .SBTTL  PRS_MNEMONIC - Parse device mnemonic
                        13BD  2939 ;+
                        13BD  2940 ; PRS_MNEMONIC  - Parse device mnemonic
                        13BD  2941 ;
                        13BD  2942 ; INPUTS:       R8      Pointer to device name
                        13BD  2943 ;               R7      Characters left in device name string
                        13BD  2944 ;
                        13BD  2945 ; OUTPUTS:      R8      Advanced by characters parsed
                        13BD  2946 ;               R7      Reduced by characters parsed
                        13BD  2947 ;               R6      Pointer to device table entry
                        13BD  2948 ;               R5      Size of network management device mnemonic
                        13BD  2949 ;               R4      Pointer to network management device mnemonic
                        13BD  2950 ;               R2      Size of VMS device mnemonic
                        13BD  2951 ;               R1      Pointer to VMS device mnemonic
                        13BD  2952 ;               R0      SS$_NORMAL    if successful
                        13BD  2953 ;                       SS$_IVDEVNAM  otherwise
                        13BD  2954 ;
                        13BD  2955 ;               All other regs are unchanged
                        13BD  2956 ;-
                        13BD  2957 PRS_MNEMONIC:
            53    DD    13BD  2958          PUSHL   R3                      ; Save reg
                        13BF  2959          ;
                        13BF  2960          ;   Parse the device mnemonic
                        13BF  2961          ;
      50  0000'8F   3C  13BF  2962          MOVZWL  #SS$_IVDEVNAM,R0        ; Assume error in name format
          54    58    D0  13C4  2963          MOVL    R8,R4                   ; Make a copy of the name pointer
          55    01    CE  13C7  2964          MNEGL   #1,R5                   ; Init the character count
                55    D6  13CA  2965 10$:     INCL    R5                      ; Update the character count
              03 57   F5  13CC  2966          SOBGTR  R7,20$                  ; If GTR then more characters are left
                0030  31  13CF  2967          BRW     100$                    ; Take common exit to report the error
                2D    88  91  13D2  2968 20$:     CMPB    (R8)+,#^A"-"        ; Mnemonic delimiter ?
                      F3  12  13D5  2969          BNEQ    10$                 ; If NEQ then keep trying
                              13D7  2970          ;
                              13D7  2971          ;   Locate the table entry
                              13D7  2972          ;
      56  0000001F'EF   9E  13D7  2973          MOVAB   DEVTRN_TABLE-DEVTRN$C_LENGTH,R6 ; Prepare to scan device table
                56    0D  C0  13DE  2974 30$:     ADDL    #DEVTRN$C_LENGTH,R6 ; Advance to next entry
                51    66  9A  13E1  2975          MOVZBL  DEVTRN$B_NETMAN(R6),R1 ; Get length of network management name
                      16  13  13E4  2976          BEQL    40$                 ; If EQL then at end of table
                55    51  D1  13E6  2977          CMPL    R1,R5               ; Same size ?
                      F3  12  13E9  2978          BNEQ    30$                 ; If NEQ no, advance to next entry
                64    55  29  13EB  2979          CMPC    R5,(R4),-           ; Is this it ?
                  01  A6      13EE  2980                      DEVTRN$T_NETMAN(R6) ;
                      EC  12  13F0  2981          BNEQ    30$                 ; If NEQ then loop
            52    06  A6  9A  13F2  2982          MOVZBL  DEVTRN$B_VMS(R6),R2 ; Setup length of VMS mnemonic
            51    07  A6  9E  13F6  2983          MOVAB   DEVTRN$T_VMS(R6),R1 ; Setup pointer to mnemonic
                      03  11  13FA  2984          BRB     50$                 ; Take common exit
                51    54  7D  13FC  2985 40$:     MOVQ    R4,R1               ; Make a copy of the Net Man. mnemonic
                50    00' D0  13FF  2986 50$:     MOVL    S^#SS$_NORMAL,R0    ; Indicate success
                      53 8ED0 1402  2987 100$:    POPL    R3                  ; Restore R3
                              1405  2988          RSB                         ; Return status in R0
```

```
                                J 4
                     1406  2990          .SBTTL  PRS_DECIMAL - Parse decimal number
                     1406  2991 ;+
                     1406  2992 ; PRS_DECIMAL   - Convert Ascii decimal number to a binary value
                     1406  2993 ;
                     1406  2994 ; INPUTS:       R7      Count of characters remaining in the string
                     1406  2995 ;               R8      Address of string
                     1406  2996 ;               R4      Field delimiter
                     1406  2997 ;
                     1406  2998 ; OUTPUTS:      R8      Advanced by characters parsed
                     1406  2999 ;               R7      Reduced by characters parsed
                     1406  3000 ;               R2      Number of numeric characters parsed
                     1406  3001 ;               R1      Binary value
                     1406  3002 ;               R0      SS$_NORMAL     if successful
                     1406  3003 ;                       SS$_INVDEVNAM  otherwise
                     1406  3004 ;-
                     1406  3005 PRS_DECIMAL:                            ; Parse/convert ascii decimal to binary
     51    7C        1406  3006          CLRQ    R1                     ; Init binary value, character count
     57    D7        1408  3007 10$:     DECL    R7                     ; Account for next character
     1E    19        140A  3008          BLSS    15$                    ; If LSS then done
50   88    9A        140C  3009          MOVZBL  (R8)+,R0               ; Get next character
50   54    91        140F  3010          CMPB    R4,R0                  ; Is it the delimiter?
     18    13        1412  3011          BEQL    20$                    ; If EQL then done
50   30    82        1414  3012          SUBB    #^A"0",R0              ; Convert to binary
     18    19        1417  3013          BLSS    30$                    ; If LSS then not decimal
09   50    91        1419  3014          CMPB    R0,#9                  ; Decimal ?
     13    1A        141C  3015          BGTRU   30$                    ; If GTRU not decimal
     52    D6        141E  3016          INCL    R2                     ; Bump number of digits
51   0A    C4        1420  3017          MULL    #10,R1                 ; Shift total one digit
     0C    1D        1423  3018          BVS     30$                    ; If VS then R1 has overflowed
51   50    C0        1425  3019          ADDL    R0,R1                  ; Complete the sum
     DE    11        1428  3020          BRB     10$                    ; Loop
     57    D4        142A  3021 15$:     CLRL    R7                     ; Repair R7
50   00'   3C        142C  3022 20$:     MOVZWL  S^#SS$_NORMAL,R0       ; Indicate success
     05    11        142F  3023          BRB     40$                    ; Take common exit
50   0000'8F  3C     1431  3024 30$:     MOVZWL  #SS$_IVDEVNAM,R0       ; Indicate invalid device name
               05    1436  3025 40$:     RSB                            ; Return status in R0
```

```
                                1437  3027          .SBTTL  DEV_CNT_QIO - Get device counters
                                1437  3028 ;+
                                1437  3029 ; DEV_CNT_QIO   - Get counters from device driver
                                1437  3030 ;
                                1437  3031 ; INPUTS:       R3      Address of next byte in output buffer
                                1437  3032 ;               R2      Channel to device driver
                                1437  3033 ;               R0      $QIO function code
                                1437  3034 ;
                                1437  3035 ; OUTPUTS:      R3      Updated to next free byte in output buffer
                                1437  3036 ;               R0      SS$_NORMAL
                                1437  3037 ;
                                1437  3038 ;               R1-R2,R4-R5 are destroyed.
                                1437  3039 ;-
                                1437  3040 DEV_CNT_QIO:
            54    53    D0      1437  3041          MOVL    R3,R4                   ; Save output buffer pointer
                  02    E1      143A  3042          BBC     #NET$V_CLRCNT,-         ; If BC, don't clear the counters
  05 00000000'EF                143C  3043                  NET$GL_FLAGS,20$        ;
     50    C000'8F    A8      1442  3044          BISW    #IO$M_CLR_COUNT,R0      ; Else clear the counters
        00000029'EF    7C      1447  3045 20$:     CLRQ    QIOW_Q_IOSB             ; Init IOSB image
51      00000000'EF    9E      144D  3046          MOVAB   TMPBUF_DESC,R1          ; Get address of TMP_BUF descriptor
55      00000033'EF    9E      1454  3047          MOVAB   TMP_Q_DESC,R5           ; Get address of our buffer descriptor
 65      04 A1    61    C1      145B  3048          ADDL3   (R1),4(R1),(R5)         ; Get ending buffer address
            65    53    C2      1460  3049          SUBL    R3,(R5)                 ; Setup bytes left in buffer
          04 A5    53    D0      1463  3050          MOVL    R3,4(R5)                ; Point to it
                                1467  3051          $QIOW_S FUNC = R0,-
                                1467  3052                  CHAN = R2,-
                                1467  3053                  EFN  = #NET$C_EFN_WAIT,-
                                1467  3054                  IOSB = QIOW_Q_IOSB,-
                                1467  3055                  P2   = R5
50      0000002B'EF    3C      1486  3056          MOVZWL  QIOW_Q_IOSB+2,R0        ; Get number of bytes returned
                  17    13      148D  3057          BEQL    90$                     ; If EQL then none
            53    50    C0      148F  3058          ADDL    R0,R3                   ; Advance R3
            50    04    D1      1492  3059          CMPL    #4,R0                   ; At least 4 bytes ?
                  0F    14      1495  3060          BGTR    90$                     ; If GTR no, return all bytes
       C000 8F    64    B1      1497  3061          CMPW    (R4),#NET$C_NMACNT_SLZ  ; Seconds since last zeroed ?
                  08    12      149C  3062          BNEQ    90$                     ; If NEQ no, return whats left
            50    04    C2      149E  3063          SUBL    #4,R0                   ; Reduce by size of counter field
 64       04 A4    50    28      14A1  3064          MOVC3   R0,4(R4),(R4)           ; Shift remaning counters, setup R3
            50    00'   3C      14A6  3065 90$:     MOVZWL  S^#SS$_NORMAL,R0        ; Indicate success
                        05      14A9  3066          RSB                             ; Done
                                14AA  3067
                                14AA  3068 .END
```

```
$$DEVTRN                = 000000EF R      02        CRI_PSI_TAB                 0000010C R      02
$$T1                    = 00000001                  CRI_TO_PSI                  00000573 R      04
$$_NSPMSG               = 00000000                  CRI_TRN_TAB                 0000016C R      02
$$_TR3MSG               = 00000000                  CRI_UNA                     00000E5E R      04
$$_TR4MSG               = 00000000                  CVT_NMA_INT                 0000133F R      04
$WIDTH_B                = 00000001                  CXB$C_OVERHEAD            = 0000004C
$WIDTH_L                = 00000003                  DC$_SCOM                  = 00000020
$WIDTH_W                = 00000002                  DEAC_PLVEC                  00000B06 R      04
ACP$C_STA_F             = 00000004                  DEFAULT_SCAN                ********    X   04
ACP$C_STA_H             = 00000005                  DEV$V_NET                 = 0000000D
ACP$C_STA_I             = 00000000                  DEVNAM_C_SIZ              = 00000020
ACP$C_STA_N             = 00000001                  DEVTRN$B_CHAR             = 0000000C
ACP$C_STA_R             = 00000002                  DEVTRN$B_DEV              = 0000000A
ACP$C_STA_S             = 00000003                  DEVTRN$B_NETMAN           = 00000000
ADJ$C_PTY_AREA          = 00000003                  DEVTRN$B_PROT             = 0000000B
ADJ$C_PTY_PH4           = 00000004                  DEVTRN$B_VMS              = 00000006
ADJ$C_PTY_PH4N          = 00000005                  DEVTRN$C_DEV_CI           = 00000004
ADJ$W_BUFSIZ            = 00000006                  DEVTRN$C_DEV_DMC          = 00000001
ADJ$W_INT_LSN           = 00000008                  DEVTRN$C_DEV_DMF          = 00000003
ADJ$W_PNA               = 00000004                  DEVTRN$C_DEV_DMP          = 00000005
ALLOC_PLVEC               000006FF R      04        DEVTRN$C_DEV_DUP          = 00000006
BIT...                  = 00000020                  DEVTRN$C_DEV_KMS          = 00000007
BUILD_DEVBUF              000012B0 R      04        DEVTRN$C_DEV_PCL          = 00000002
CCB$L_UCB               = 00000000                  DEVTRN$C_DEV_PPUNA        = 0000000A
CLEAR_VOLATILE            00000360 R      04        DEVTRN$C_DEV_UNA          = 00000009
CNF$B_FLG               = 0000000B                  DEVTRN$C_DEV_UNK          = 00000000
CNF$CLR_FIELD             ********    X   04        DEVTRN$C_DEV_X25          = 00000008
CNF$GET_FIELD             ********    X   04        DEVTRN$C_LENGTH           = 0000000D
CNF$KEY_SEARCH            ********    X   04        DEVTRN$M_MULTI            = 00000001
CNF$PUT_FIELD             ********    X   04        DEVTRN$T_NETMAN           = 00000001
CNF$V_FLG_MRK1          = 00000004                  DEVTRN$T_VMS              = 00000007
CNF$W_ID                = 00000012                  DEVTRN$V_MULTI            = 00000000
CNF$_ADVANCE            = 00000000                  DEVTRN_TABLE                0000002C R      02
CNF$_QUIT               = 00000002                  DEV_CNT_QIO                 00001437 R      04
CNF$_TAKE_CURR          = 00000003                  DLLQIO$C_LENGTH           = 00000018
CNF$_TAKE_PREV          = 00000001                  DLLQIO$L_FUNC             = 00000000
CNR$B_TYPE              = 0000000A                  DLLQIO$L_P1               = 00000004
CNVTAB$C_INTRNL         = 00003FFF                  DLLQIO$L_P2               = 00000008
CNVTAB$M_FMT            = C0000000                  DLLQIO$L_P3               = 0000000C
CNVTAB$M_INT            = 3FFF0000                  DT$_DMC1T                 = 00000001
CNVTAB$M_NMA            = 00000FFF                  DVI$_DEVCHAR              = 00000002
CNVTAB$M_RO             = 00001000                  DVI$_DEVNAM               = 00000020
CNVTAB$S_FMT            = 00000002                  EVC$C_SRC_CIR             = 00000003
CNVTAB$S_INT            = 0000000E                  EVC$C_SRC_LIN             = 00000001
CNVTAB$S_NMA            = 0000000C                  GET_PSI_CRI                 00000281 R      04
CNVTAB$S_RO             = 00000001                  GET_PSI_PLI                 000002C7 R      04
CNVTAB$V_FMT            = 0000001E                  IO$M_CLR_COUNT              ********    X   04
CNVTAB$V_INT            = 00000010                  IO$M_CTRL                   ********    X   04
CNVTAB$V_NMA            = 00000000                  IO$M_RD_COUNT               ********    X   04
CNVTAB$V_RO             = 0000000C                  IO$M_SHUTDOWN               ********    X   04
CRI_CI                    00000E5E R      04        IO$M_STARTUP                ********    X   04
CRI_CLR_TAB               00000104 R      02        IO$_ACPCONTROL              ********    X   04
CRI_DMC                   00000E62 R      04        IO$_SENSEMODE               ********    X   04
CRI_DMP                   00000E5E R      04        IO$_SETMODE                 ********    X   04
CRI_FOREIGN               00000E5E R      04        IOC$VERIFYCHAN              ********    X   04
CRI_PCL                   00000E5E R      04        JAM_CNF                     000001E8 R      04
CRI_PSI_SIZ             = 00000020                  LOG_COUNTERS                ********    X   04
```

```
LPD$B_ASTCNT                =  0000001B
LPD$B_CNT_IFL               =  0000004F
LPD$B_CNT_LDN               =  0000004E
LPD$B_PLVEC                 =  00000028
LPD$B_PTH_INX               =  00000020
LPD$B_SUB_STA               =  00000027
LPD$C_CNT_SIZE              =  00000016
LPD$L_ABS_TIM               =  00000036
LPD$L_CNT_APR               =  0000003A
LPD$L_CNT_DPS               =  00000042
LPD$L_CNT_TPR               =  0000003E
LPD$L_CNT_TPS               =  00000046
LPD$L_UCB                   =  00000010
LPD$V_BC                    =  0000000A
LPD$V_DLE                   =  00000002
LPD$V_RUN                   =  00000004
LPD$V_X25                   =  00000007
LPD$W_CHAN                  =  00000014
LPD$W_CNT_ACL               =  0000004A
LPD$W_CNT_TCL               =  0000004C
LPD$W_DRT                   =  0000002C
LPD$W_STS                   =  00000022
LPD_CNT_TAB                    00000000 R      02
MAX_C_DEVNAM                =  0000000F
MOVE_FMT_CNT                   ********   X    04
NET$ALLOCATE                   ********   X    04
NET$ALONPAGED                  ********   X    04
NET$APPLY_DFLT                 ********   X    04
NET$BIN2ASC                    ********   X    04
NET$CRI_L_BLO                  00000B89 RG     04
NET$CRI_L_DRT                  00000B5D RG     04
NET$CRI_L_LIT                  00000BAE RG     04
NET$CRI_L_MST                  00000C05 RG     04
NET$CRI_L_PNA                  00000BD8 RG     04
NET$CRI_L_SUB                  00000B48 RG     04
NET$CRI_S_CHR                  00000E12 RG     04
NET$CRI_S_CNT                  00000D3F RG     04
NET$CRI_S_COL                  00000C1A RG     04
NET$CRI_S_DEVNAM               00000CDA RG     04
NET$CRI_S_LOO                  00000C81 RG     04
NET$CRI_S_PNN                  00000C34 RG     04
NET$CRI_S_VMSNAM               00000CC6 RG     04
NET$CRI_V_LCK                  00000B34 RG     04
NET$CVT_NMA_INT                00001324 RG     04
NET$C_ACT_TIMER             =  0000001E
NET$C_EFN_ASYN              =  00000002
NET$C_EFN_WAIT              =  00000001
NET$C_IPL                   =  00000008
NET$C_MAXACCFLD             =  00000027
NET$C_MAXLINNAM             =  0000000F
NET$C_MAXLNK                =  000003FF
NET$C_MAXNODNAM             =  00000006
NET$C_MAXOBJNAM             =  0000000C
NET$C_MAX_AREAS             =  0000003F
NET$C_MAX_LINES             =  00000040
NET$C_MAX_NCB               =  0000006E
NET$C_MAX_NODES             =  000003FF
NET$C_MAX_OBJ               =  000000FF
NET$C_MAX_WQE               =  00000014
NET$C_MINBUFSIZ             =  000000C0
NET$C_NMACNT_SLZ            =  0000C000
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$DEALLOCATE                 ********   X    04
NET$DECLARE_PSI                ********   X    04
NET$DEFAULT_CRI                0000036B RG     04
NET$DEFAULT_PLI                00000420 RG     04
NET$DELETE_CRI                 00000A78 RG     04
NET$DELETE_PLI                 00000AC6 RG     04
NET$DLL_OPR_SET                ********   X    04
NET$GET_PLVECLFD               ********   X    04
NET$GET_X25_CHAN               ********   X    04
NET$GL_CNR_CNI                 ********   X    04
NET$GL_CNR_NDI                 ********   X    04
NET$GL_CNR_PLI                 ********   X    04
NET$GL_FLAGS                   ********   X    04
NET$GL_PTR_LNI                 ********   X    04
NET$GL_PTR_VCB                 ********   X    04
NET$GL_SRCH2_ID                ********   X    04
NET$GL_SRCH_ID                 ********   X    04
NET$GQ_SRCH2_KEY               ********   X    04
NET$GQ_SRCH_KEY                ********   X    04
NET$GW_X25_CHAN                ********   X    04
NET$G_CRI_DDCMP                ********   X    04
NET$G_CRI_DLM                  ********   X    04
NET$G_CRI_DLMOUT               ********   X    04
NET$G_CRI_NI                   ********   X    04
NET$G_CRI_TRN                  ********   X    04
NET$G_CRI_X25                  ********   X    04
NET$G_PLI_DDCMP                ********   X    04
NET$G_PLI_LAPB                 ********   X    04
NET$G_PLI_NI                   ********   X    04
NET$INSERT_CRI                 000004BC RG     04
NET$INSERT_PLI                 000005EF RG     04
NET$JNX_CO                     ********   X    04
NET$LOCATE_LPD                 ********   X    04
NET$M_MAXLNKMSK             =  000003FF
NET$NDI_BY_ADD                 ********   X    04
NET$PLI_L_BUS                  00000EE2 RG     04
NET$PLI_L_PLVEC                00000EDA RG     04
NET$PLI_L_SUB                  00000ED7 RG     04
NET$PLI_S_CHR                  0000107F RG     04
NET$PLI_S_CNT                  00000FC7 RG     04
NET$PLI_S_COL                  00000F26 RG     04
NET$PLI_S_DEVNAM               00000F60 RG     04
NET$PLI_S_VMSNAM               00000F40 RG     04
NET$PLI_V_LCK                  00000EB9 RG     04
NET$PRE_QIO_CRI                00000003 RG     04
NET$PRE_QIO_PLI                0000000C RG     04
NET$REMOVE_CRI                 00000B2D RG     04
```

```
NET$REMOVE_DEF                  ********    X   04
NET$REMOVE_PLI                  00000B2D RG     04
NET$SCAN_CRI                    00000000 RG     04
NET$SCAN_PLI                    00000000 RG     04
NET$SET_CTR_TIMER               ********    X   04
NET$SET_QIOW                    00000844 RG     04
NET$SHOW_CRI                    00000118 RG     04
NET$SHOW_PLI                    0000015C RG     04
NET$TABLE_DFLT                  ********    X   04
NET$V_CLRCNT                =   00000002
NET$V_CNFLCK                =   0000000B
NFB$B_DATABASE              =   00000002
NFB$B_FCT                   =   00000000
NFB$B_FLAGS                 =   00000001
NFB$B_OPER                  =   00000003
NFB$C_CRI_ACB               =   04010029
NFB$C_CRI_ACI               =   0401002A
NFB$C_CRI_BBT               =   04010025
NFB$C_CRI_CHN               =   04010021
NFB$C_CRI_CNT               =   04020044
NFB$C_CRI_DLM               =   04000005
NFB$C_CRI_DTE               =   04020049
NFB$C_CRI_DTH               =   04010031
NFB$C_CRI_DYB               =   0401002E
NFB$C_CRI_DYI               =   0401002F
NFB$C_CRI_DYT               =   04010030
NFB$C_CRI_IAB               =   0401002B
NFB$C_CRI_IAI               =   0401002C
NFB$C_CRI_IAT               =   0401002D
NFB$C_CRI_MBL               =   04010022
NFB$C_CRI_MRB               =   04010027
NFB$C_CRI_MST               =   04010032
NFB$C_CRI_MTR               =   04010028
NFB$C_CRI_MWI               =   04010023
NFB$C_CRI_NAM               =   04020041
NFB$C_CRI_PLS               =   0401001E
NFB$C_CRI_POL               =   0401001D
NFB$C_CRI_SER               =   04000002
NFB$C_CRI_STA               =   04010013
NFB$C_CRI_TRI               =   04010024
NFB$C_CRI_TRT               =   04010026
NFB$C_CRI_TYP               =   04010020
NFB$C_CRI_USE               =   0401001F
NFB$C_CRI_VMSNAM            =   04020042
NFB$C_CTX_SIZE              =   00000040
NFB$C_DB_CRI                =   00000004
NFB$C_ENDOFLIST             =   00000000
NFB$C_FC_DELETE             =   00000021
NFB$C_FC_SET                =   00000023
NFB$C_FC_SHOW               =   00000022
NFB$C_FC_ZERCOU             =   00000025
NFB$C_LENGTH                =   00000010
NFB$C_NDI_NLI               =   0202004C
NFB$C_NDI_NNA               =   02020043
NFB$C_OP_EQL                =   00000000
NFB$C_PLI_BFN               =   0501001E
NFB$C_PLI_BFS               =   05010027
NFB$C_PLI_BUS               =   0501001F
NFB$C_PLI_CHR               =   05020043
NFB$C_PLI_CLO               =   05000005
NFB$C_PLI_CNT               =   05020044
NFB$C_PLI_CON               =   05000004
NFB$C_PLI_DDT               =   0501001B
NFB$C_PLI_DLT               =   0501001C
NFB$C_PLI_DUP               =   05000003
NFB$C_PLI_EPT               =   05010026
NFB$C_PLI_HTI               =   05010016
NFB$C_PLI_HWA               =   05020046
NFB$C_PLI_MBL               =   05010017
NFB$C_PLI_MCD               =   05020045
NFB$C_PLI_MOD               =   05010022
NFB$C_PLI_MRT               =   05010018
NFB$C_PLI_MWI               =   05010019
NFB$C_PLI_NAM               =   05020041
NFB$C_PLI_PLVEC             =   05010020
NFB$C_PLI_PRO               =   05010014
NFB$C_PLI_RTT               =   05010021
NFB$C_PLI_SLT               =   0501001A
NFB$C_PLI_SRT               =   0501001D
NFB$C_PLI_STA               =   05010011
NFB$C_PLI_SUB               =   05010012
NFB$C_TYP_STR               =   00000002
NFB$L_FLDID                 =   00000010
NFB$L_SRCH_KEY              =   00000004
NFB$M_INX                   =   0000FFFF
NFB$M_NOCTX                 =   00000004
NFB$M_TYP                   =   00030000
NFB$S_TYP                   =   00000002
NFB$V_DB                    =   00000018
NFB$V_INX                   =   00000000
NFB$V_TYP                   =   00000010
NFB_SIZE                    =   000000B8
NMA$C_ACC_EXC               =   00000003
NMA$C_ACC_LIM               =   00000002
NMA$C_CIRTY_X25             =   00000003
NMA$C_CIRUS_INC             =   00000001
NMA$C_CIRUS_OUT             =   00000002
NMA$C_CIRUS_PER             =   00000000
NMA$C_CTCIR_ACL             =   00000322
NMA$C_CTCIR_APR             =   00000320
NMA$C_CTCIR_DPS             =   00000321
NMA$C_CTCIR_IFL             =   00000335
NMA$C_CTCIR_LDN             =   00000334
NMA$C_CTCIR_TCL             =   0000032C
NMA$C_CTCIR_TPR             =   0000032A
NMA$C_CTCIR_TPS             =   0000032B
NMA$C_LINMC_SET             =   00000001
NMA$C_LINPR_                =   FFFFFFFF
NMA$C_LINPR_CON             =   00000001
NMA$C_LINPR_DMC             =   00000004
NMA$C_LINPR_LAPB            =   00000005
NMA$C_LINPR_MAS             =   00000001
NMA$C_LINPR_NI              =   00000006
NMA$C_LINPR_POI             =   00000000
```

```
NMA$C_LINPR_TRI                 = 00000002
NMA$C_PCCI_ACB                  = 0000047E
NMA$C_PCCI_ACI                  = 0000047F
NMA$C_PCCI_BBT                  = 00000475
NMA$C_PCCI_DTH                  = 00000486
NMA$C_PCCI_DYB                  = 00000483
NMA$C_PCCI_DYI                  = 00000484
NMA$C_PCCI_DYT                  = 00000485
NMA$C_PCCI_IAB                  = 00000480
NMA$C_PCCI_IAI                  = 00000481
NMA$C_PCCI_IAT                  = 00000482
NMA$C_PCCI_MRB                  = 00000479
NMA$C_PCCI_MST                  = 00000AFA
NMA$C_PCCI_MTR                  = 0000047A
NMA$C_PCCI_PLS                  = 000003F3
NMA$C_PCCI_POL                  = 000003F2
NMA$C_PCCI_TRI                  = 00000474
NMA$C_PCCI_TRT                  = 00000476
NMA$C_PCCI_TYP                  = 00000458
NMA$C_PCLI_ACC                  = 00000B1E
NMA$C_PCLI_BFN                  = 00000451
NMA$C_PCLI_BUS                  = 00000AF1
NMA$C_PCLI_CLO                  = 00000459
NMA$C_PCLI_CON                  = 00000456
NMA$C_PCLI_CRC                  = 00000B1C
NMA$C_PCLI_DCH                  = 00000B1B
NMA$C_PCLI_DDT                  = 0000047F
NMA$C_PCLI_DES                  = 00000B21
NMA$C_PCLI_DLT                  = 00000480
NMA$C_PCLI_DUP                  = 00000457
NMA$C_PCLI_HWA                  = 00000488
NMA$C_PCLI_MCA                  = 00000B0F
NMA$C_PCLI_MLT                  = 00000B19
NMA$C_PCLI_PAD                  = 00000B1A
NMA$C_PCLI_PHA                  = 00000B04
NMA$C_PCLI_PRM                  = 00000B18
NMA$C_PCLI_PRO                  = 00000458
NMA$C_PCLI_PTY                  = 00000B0E
NMA$C_PCLI_RTT                  = 00000461
NMA$C_PCLI_SLT                  = 0000047E
NMA$C_PCLI_SRT                  = 00000481
NMA$C_STATE_OFF                 = 00000001
NMA$C_STATE_ON                  = 00000000
NMA$C_STATE_SER                 = 00000002
NSP$$$_QUAL_ACK                 = 00000000
NSP$$$_QUAL_ALTFLW              = 00000000
NSP$$$_QUAL_DATA                = 00000000
NSP$$$_QUAL_FLW                 = 00000000
NSP$$$_QUAL_INF                 = 00000000
NSP$$$_QUAL_MSG                 = 00000000
NSP$$$_QUAL_SRV                 = 00000000
NSP$C_EXT_LNK                   = 0000001E
NSP$C_FLW_DATA                  = 00000000
NSP$C_FLW_INT                   = 00000001
NSP$C_FLW_NOP                   = 00000000
NSP$C_FLW_XOFF                  = 00000001
NSP$C_FLW_XON                   = 00000002
NSP$C_HSZ_ACK                   = 00000007
NSP$C_HSZ_CA                    = 00000003
NSP$C_HSZ_CC                    = 00000064
NSP$C_HSZ_CD                    = 000000F0
NSP$C_HSZ_CI                    = 000000F0
NSP$C_HSZ_DATA                  = 00000009
NSP$C_HSZ_DC                    = 00000016
NSP$C_HSZ_DI                    = 00000016
NSP$C_HSZ_INT                   = 00000009
NSP$C_HSZ_LS                    = 00000009
NSP$C_INF_V31                   = 00000001
NSP$C_INF_V32                   = 00000000
NSP$C_INF_V33                   = 00000002
NSP$C_MAXADR                    = 00000009
NSP$C_MSG_CA                    = 00000024
NSP$C_MSG_CC                    = 00000028
NSP$C_MSG_CI                    = 00000018
NSP$C_MSG_DATA                  = 00000000
NSP$C_MSG_DC                    = 00000048
NSP$C_MSG_DI                    = 00000038
NSP$C_MSG_DTACK                 = 00000004
NSP$C_MSG_INT                   = 00000030
NSP$C_MSG_LIACK                 = 00000014
NSP$C_MSG_LS                    = 000C0010
NSP$C_SRV_MFC                   = 00000002
NSP$C_SRV_NFC                   = 00000000
NSP$C_SRV_REQ                   = 00000001
NSP$C_SRV_SFC                   = 00000001
NSP$M_ACK_NAK                   = 00001000
NSP$M_ACK_NUM                   = 00000FFF
NSP$M_ACK_VALID                 = 00008000
NSP$M_DATA_BOM                  = 00000020
NSP$M_DATA_EOM                  = 00000040
NSP$M_DATA_OVFW                 = 00000080
NSP$M_FLW_CHAN                  = 0000000C
NSP$M_FLW_DRV                   = 000000F0
NSP$M_FLW_INT                   = 00000020
NSP$M_FLW_INUSE                 = 00000010
NSP$M_FLW_LISUB                 = 00000004
NSP$M_FLW_MODE                  = 00000003
NSP$M_FLW_SP1                   = 00000008
NSP$M_FLW_SP2                   = 00000040
NSP$M_FLW_SP3                   = 00000080
NSP$M_FLW_XOFF                  = 00000001
NSP$M_FLW_XON                   = 00000002
NSP$M_INF_VER                   = 00000003
NSP$M_MSG_INT                   = 00000020
NSP$M_MSG_LI                    = 00000010
NSP$M_SRV_01                    = 00000003
NSP$M_SRV_EXT                   = 00000080
NSP$M_SRV_FLW                   = 0000000C
NSP$M_SRV_REQ                   = 000000F3
NSP$M_SRV_SP1                   = 00000070
NSP$R_QUAL                      = 00000000
NSP$S_ACK_NUM                   = 0000000C
NSP$S_ACK_SP2                   = 00000002
NSP$S_DATA_SP                   = 00000005
```

```
NSP$S_FLW_CHAN          = 00000002
NSP$S_FLW_DRV           = 00000004
NSP$S_FLW_MODE          = 00000002
NSP$S_INF_VER           = 00000002
NSP$S_MSG_SP1           = 00000004
NSP$S_NSPMSG            = 00000005
NSP$S_QUAL              = 00000005
NSP$S_QUAL_ACK          = 00000002
NSP$S_QUAL_ALTFLW       = 00000001
NSP$S_QUAL_DATA         = 00000001
NSP$S_QUAL_FLW          = 00000001
NSP$S_QUAL_INF          = 00000001
NSP$S_QUAL_MSG          = 00000005
NSP$S_QUAL_SRV          = 00000001
NSP$S_SRV_01            = 00000002
NSP$S_SRV_FLW           = 00000002
NSP$S_SRV_SP1           = 00000003
NSP$V_ACK_NAK           = 0000000C
NSP$V_ACK_NUM           = 00000000
NSP$V_ACK_SP2           = 0000000D
NSP$V_ACK_VALID         = 0000000F
NSP$V_DATA_BOM          = 00000005
NSP$V_DATA_EOM          = 00000006
NSP$V_DATA_OVFW         = 00000007
NSP$V_DATA_SP           = 00000000
NSP$V_FLW_CHAN          = 00000002
NSP$V_FLW_DRV           = 00000004
NSP$V_FLW_INT           = 00000005
NSP$V_FLW_INUSE         = 00000004
NSP$V_FLW_LISUB         = 00000002
NSP$V_FLW_MODE          = 00000000
NSP$V_FLW_SP1           = 00000003
NSP$V_FLW_SP2           = 00000006
NSP$V_FLW_SP3           = 00000007
NSP$V_FLW_XOFF          = 00000000
NSP$V_FLW_XON           = 00000001
NSP$V_INF_VER           = 00000000
NSP$V_MSG_INT           = 00000005
NSP$V_MSG_LI            = 00000004
NSP$V_MSG_SP1           = 00000000
NSP$V_SRV_01            = 00000000
NSP$V_SRV_EXT           = 00000007
NSP$V_SRV_FLW           = 00000002
NSP$V_SRV_SP1           = 00000004
NSP$W_DSTLNK            = 00000001
NSP$W_SRCLNK            = 00000003
P2BUF_SIZE              = 00000066
P4BUF_SIZE              = 00000200
PLI_B_STATE               00000000 R  03
PLI_CI                    000010DF R  04
PLI_CLR_TAB               000000FC R  02
PLI_CNT_TAB               00000028 R  02
PLI_DMC                   0000125D R  04
PLI_DMP                   000010DF R  04
PLI_FOREIGN               000010DF R  04
PLI_PCL                   000010DF R  04
PLI_PPUNA                 0000119B R  04
PLI_PPUNA_TAB             0000118B R  04
PLI_PSI_SIZ             = 00000038
PLI_PSI_TAB               00000134 R  02
PLI_Q_DEVNAM              00000001 R  03
PLI_TO_PSI                000008F1 R  04
PLI_TRN_TAB               000001B8 R  02
PLI_UNA                   000010E3 R  04
PLI_V_NEW               = 00000004
PLVEC$AB_DEV              ******** X  04
PLVEC$AB_REFC             ******** X  04
PLVEC$AB_STATE            ******** X  04
PLVEC$AL_ABS_TIM          ******** X  04
PLVEC$AL_UCB              ******** X  04
PLVEC$AW_CHAN             ******** X  04
PLVEC$GB_MAX              ******** X  04
PREQIO_PRI_CRI            00000013 R  04
PRS_DECIMAL               00001406 R  04
PRS_MNEMONIC              000013BD R  04
PSI_PLI_CLR_TAB           0000012C R  02
QIOW_Q_IOSB               00000029 R  03
QIOW_W_RETLEN             00000031 R  03
RCB$B_ETY               = 0000008A
RCB$B_HOMEAREA          = 0000008B
RCB$L_PTR_ADJ           = 0000002C
RCB$W_ADDR              = 0000000E
RCB$W_TOTBUFSIZ         = 0000007E
REBUILD_NAME              0000007B R  04
SEND_TO_PSI               0000093B R  04
SHO_PLI_CRI               0000019E R  04
SIZ...                  = 00000002
SRCH_BUF                  00000009 R  03
SS$_BADPARAM              ******** X  04
SS$_INSFARG               ******** X  04
SS$_INSFMEM               ******** X  04
SS$_IVDEVNAM              ******** X  04
SS$_NORMAL                ******** X  04
SS$_NOSUCHDEV             ******** X  04
SS$_RESULTOVF             ******** X  04
SS$_WRITLCK               ******** X  04
STORE_PSI_PARAMS          000002F4 R  04
SUPPRESS_AREA             ******** X  04
SYS$ASSIGN                ******** GX 04
SYS$DASSGN                ******** GX 04
SYS$GETDVI                ******** GX 04
SYS$QIOW                  ******** GX 04
TAKE_2_IOSB               00000270 R  04
TAKE_R7_IOSB              00000273 R  04
TMPBUF_DESC               ******** X  04
TMP_Q_DESC                00000033 R  03
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_PRT_ECL            = 0000001F
```

```
TR$C_PRI_RTHRU                  =  0000001F
TR3$$$_QUAL_MSG                 =  00000000
TR3$$$_QUAL_RTFLG               =  00000000
TR3$C_HSZ_DATA                  =  00000006
TR3$C_MSG_DATA                  =  00000002
TR3$C_MSG_HELLO                 =  00000005
TR3$C_MSG_INIT                  =  00000001
TR3$C_MSG_NOP2                  =  00000008
TR3$C_MSG_ROUT                  =  00000007
TR3$C_MSG_STR2                  =  00000058
TR3$C_MSG_VERF                  =  00000003
TR3$M_MSG_CTL                   =  00000001
TR3$M_MSG_RTH                   =  00000002
TR3$M_RTFLG_PH2                 =  00000040
TR3$M_RTFLG_RQR                 =  00000008
TR3$M_RTFLG_RTS                 =  00000010
TR3$R_QUAL                      =  00000000
TR3$S_QUAL                      =  00000001
TR3$S_QUAL_MSG                  =  00000001
TR3$S_QUAL_RTFLG                =  00000001
TR3$S_RTFLG_012                 =  00000003
TR3$S_TR3MSG                    =  00000001
TR3$V_MSG_CTL                   =  00000000
TR3$V_MSG_RTH                   =  00000001
TR3$V_RTFLG_012                 =  00000000
TR3$V_RTFLG_5                   =  00000005
TR3$V_RTFLG_7                   =  00000007
TR3$V_RTFLG_PH2                 =  00000006
TR3$V_RTFLG_RQR                 =  00000003
TR3$V_RTFLG_RTS                 =  00000004
TR4$$$_QUAL_ADDR                =  00000000
TR4$$$_QUAL_RTFLG               =  00000000
TR4$$$_QUAL_SCLASS              =  00000000
TR4$C_BCE_MID1                  =  040000AB
TR4$C_BCE_MID2                  =  00000000
TR4$C_BCR_MID1                  =  030000AB
TR4$C_BCR_MID2                  =  00000000
TR4$C_BCT3MULT                  =  00000008
TR4$C_END_NODE                  =  00000003
TR4$C_HIORD                     =  000400AA
TR4$C_HSZ_DATA                  =  00000015
TR4$C_MSG_BCEHEL                =  0000000D
TR4$C_MSG_BCRHEL                =  0000000B
TR4$C_MSG_LDATA                 =  00000006
TR4$C_MSG_RDATA                 =  00000002
TR4$C_PRO_TYPE                  =  00000360
TR4$C_RTR_LVL1                  =  00000002
TR4$C_RTR_LVL2                  =  00000001
TR4$C_T3MULT                    =  00000002
TR4$C_VER_HIB                   =  00000000
TR4$C_VER_LOWW                  =  00000002
TR4$M_ADDR_AREA                 =  0000FC00
TR4$M_ADDR_DEST                 =  000003FF
TR4$M_RTFLG_INI                 =  00000020
TR4$M_RTFLG_LNG                 =  00000004
TR4$M_RTFLG_RQR                 =  00000008
TR4$M_RTFLG_RTS                 =  00000010
TR4$R_QUAL                      =  00000000
TR4$S_ADDR_AREA                 =  00000006
TR4$S_ADDR_DEST                 =  0000000A
TR4$S_QUAL                      =  00000002
TR4$S_QUAL_ADDR                 =  00000002
TR4$S_QUAL_RTFLG                =  00000001
TR4$S_QUAL_SCLASS               =  00000001
TR4$S_RTFLG_01                  =  00000002
TR4$S_RTFLG_VER                 =  00000002
TR4$S_SCLASS_57                 =  00000003
TR4$S_TR4MSG                    =  00000002
TR4$V_ADDR_AREA                 =  0000000A
TR4$V_ADDR_DEST                 =  00000000
TR4$V_RTFLG_01                  =  00000000
TR4$V_RTFLG_INI                 =  00000005
TR4$V_RTFLG_LNG                 =  00000002
TR4$V_RTFLG_RQR                 =  00000003
TR4$V_RTFLG_RTS                 =  00000004
TR4$V_RTFLG_VER                 =  00000006
TR4$V_SCLASS_1                  =  00000001
TR4$V_SCLASS_57                 =  00000005
TR4$V_SCLASS_BC                 =  00000004
TR4$V_SCLASS_LS                 =  00000002
TR4$V_SCLASS_METR               =  00000000
TR4$V_SCLASS_SUBA               =  00000003
TRAN_DEVNAM                       0000136C  R      04
UCB$B_DEVCLASS                  =  00000040
UCB$B_DEVTYPE                   =  00000041
UCB$L_DEVDEPEND                 =  00000044
UCB$W_DEVBUFSIZ                 =  00000042
X25_CNT_TAB                       00000024  R      02
XM$V_CHR_HDPLX                  =  00000002
XM$V_CHR_LOOPB                  =  00000001
XM$V_CHR_MOP                    =  00000000
XM$V_STS_ACTIVE                 =  0000000B
_$$_                            =  000000EF
```

E 5

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_PURE	000001EC (492.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
NET_IMPURE	0000003B (59.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
NET_CODE	000014AA (5290.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.09	00:00:00.86
Command processing	136	00:00:00.98	00:00:03.49
Pass 1	1392	00:00:40.45	00:01:21.84
Symbol table sort	15	00:00:05.13	00:00:07.36
Pass 2	1392	00:00:10.20	00:00:24.47
Symbol table output	59	00:00:00.52	00:00:00.68
Psect synopsis output	2	00:00:00.05	00:00:00.16
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	3030	00:00:57.42	00:01:58.87

The working set limit was 1500 pages.
224419 bytes (439 pages) of virtual memory were used to buffer the intermediate code.
There were 190 pages of symbol table space allocated to hold 3313 non-local and 227 local symbols.
3068 source lines were read in Pass 1, producing 44 object records in Pass 2.
69 pages of virtual memory were used to define 57 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]NMALIBRY.MLB;1	1
_255DUA28:[SHRLIB]EVCDEF.MLB;1	1
_255DUA28:[NETACP.OBJ]NETDRV.MLB;1	1
_255DUA28:[NETACP.OBJ]NET.MLB;1	18
_255DUA28:[SYS.OBJ]LIB.MLB;1	3
_255DUA28:[SYSLIB]STARLET.MLB;2	18
TOTALS (all libraries)	42

3516 GETS were required to define 42 macros.

There were no errors, warnings or information messages.

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

NETCNF
LIS

NETCNFACT
LIS

NETCNFDLL
LIS

NETDLE
LIS

NETCONFIG
LIS

NETCONECT
LIS

NETCTLALL
LIS