

NNN	NNN	EEEEEEEEE	TTTTTTTTT	AAAAAAA	CCCCCCC	PPPPPPP	
NNN	NNN	EEEEEEEEE	TTTTTTTTT	AAAAAAA	CCCCCCC	PPPPPPP	
NNN	NNN	EEEEEEEEE	TTTTTTTTT	AAAAAAA	CCCCCCC	PPPPPPP	
NNN	NNN	EEE	TTT	AAA	CCC	PPP	
NNN	NNN	EEE	TTT	AAA	CCC	PPP	
NNN	NNN	EEE	TTT	AAA	CCC	PPP	
NNNNNN	NNN	EEE	TTT	AAA	CCC	PPP	
NNNNNN	NNN	EEE	TTT	AAA	CCC	PPP	
NNNNNN	NNN	EEE	TTT	AAA	CCC	PPP	
NNNNNN	NNN	EEE	TTT	AAA	CCC	PPP	
NNN	NNN	NNN	EEEEEEEEE	TTT	AAA	CCC	PPPPPPP
NNN	NNN	NNN	EEEEEEEEE	TTT	AAA	CCC	PPPPPPP
NNN	NNN	NNN	EEEEEEEEE	TTT	AAA	CCC	PPPPPPP
NNN	NNNNNN	EEE	TTT	AAAAAAA	CCC	PPP	
NNN	NNNNNN	EEE	TTT	AAAAAAA	CCC	PPP	
NNN	NNNNNN	EEE	TTT	AAAAAAA	CCC	PPP	
NNN	NNN	EEE	TTT	AAA	CCC	PPP	
NNN	NNN	EEE	TTT	AAA	CCC	PPP	
NNN	NNN	EEE	TTT	AAA	CCC	PPP	
NNN	NNN	EEEEEEEEE	TTT	AAA	CCCCCCC	PPP	
NNN	NNN	EEEEEEEEE	TTT	AAA	CCCCCCC	PPP	
NNN	NNN	EEEEEEEEE	TTT	AAA	CCCCCCC	PPP	

NE

NE

SR

S
Ps
--
NE

FILEID**NETCNFDLL

E 15

NE
VO

NN	NN	EEEEEEEEE	TTTTTTTTT	CCCCCCC	NN	NN	FFFFFFF	DDDDDDD	LL	LL
NN	NN	EEEEEEEEE	TTTTTTTTT	CCCCCCC	NN	NN	FFF	DDDDDDDD	LL	LL
NN	NN	EE	TT	CC	NN	NN	FF	DD	DD	LL
NN	NN	EE	TT	CC	NNNN	NNNN	FF	DD	DD	LL
NNNN	NN	EE	TT	CC	NNNN	NNNN	FF	DD	DD	LL
NNNN	NN	EE	TT	CC	NNNN	NNNN	FF	DD	DD	LL
NN	NN	NN	EEEEEEE	TT	CC	NN	NN	FFFF	DD	LL
NN	NN	NN	EEEEEEE	TT	CC	NN	NN	FFFF	DD	LL
NN	NNNN	EE	TT	CC	NN	NNNN	FF	DD	DD	LL
NN	NNNN	EE	TT	CC	NN	NNNN	FF	DD	DD	LL
NN	NN	EE	TT	CC	NN	NN	FF	DD	DD	LL
NN	NN	EE	TT	CC	NN	NN	FF	DD	DD	LL
NN	NN	EEEEEEEEE	TT	CCCCCCC	NN	NN	FF	DDDDDDDD	LLLLLLL	LLLLLLL
NN	NN	EEEEEEEEE	TT	CCCCCCC	NN	NN	FF	DDDDDDDD	LLLLLLL	LLLLLLL

LL	IIIIII	SSSSSSS
LL	IIIIII	SSSSSSS
LL	II	SS
LLLLLLL	IIIIII	SSSSSSS
LLLLLLL	IIIIII	SSSSSSS

(2)	245	DECLARATIONS
(7)	561	NET\$SCAN xxx - Scan database
(8)	602	NET\$PRE_QIO_xxx - Pre-QIO processing
(9)	767	NET\$SHOW xxx - Pre-SHOW processing
(10)	861	JAM CNF = Store driver values into CNF
(11)	960	GET-PSI xxx - Get current PSI parameter values
(12)	1065	CLEAR_VOLATILE - Clear list of volatile parameters
(13)	1089	NET\$DEFAULT_CRI - Apply default values
(14)	1170	NET\$DEFAULT_PLI - Apply default values
(15)	1252	NET\$INSERT_CRI - Pre-insertion processing
(16)	1339	CRI_TO_PSI - Send CRI parameters to PSIACP
(17)	1409	NET\$INSERT_PLI - Pre-insertion processing
(18)	1525	ALLOC_PLVEC - Setup PLVEC entry for new line
(19)	1631	NET\$SET_QIOW - Issue datalink SETMODE function
(20)	1699	PLI_TO_PSI - Send PLI parameters to PSIACP
(21)	1741	SEND_TO_PSI - Send control QIO to PSIACP
(22)	1897	NET\$DELETE_CRI - Pre-delete processing
(23)	1945	NET\$DELETE_PLI - Pre-delete processing
(24)	1994	NET\$REMOVE_xxx - Pre-remove processing
(25)	2012	CRI parameter action routines
(26)	2374	PLI parameter action routines
(27)	2750	BUILD_DEVBUF - Build DLLQIO buffer
(28)	2813	NET\$CVT_NMA_INT - Convert NMA to NFB code
(29)	2872	TRAN_DEVNAM - Translate device name
(30)	2938	PRS_MNEMONIC - Parse device mnemonic
(31)	2990	PRS_DECIMAL - Parse decimal number
(32)	3027	DEV_CNT_QIO - Get device counters

```
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 : FACILITY: NETWORK ACP
0000 29
0000 30 : ABSTRACT:
0000 31 : Action routines for CRI and PLI databases.
0000 32
0000 33 : This module provides support to the NETACP database management
0000 34 : including database entry insertion and action routines to
0000 35 : retrieve data for parameters which are not stored in any of
0000 36 : the datalink CNF control blocks (CRI and PLI).
0000 37
0000 38 : ENVIRONMENT:
0000 39 : The module runs in kernel mode and at possibly elevated IPL.
0000 40 : It is therefore locked into the ACP's virtual address space
0000 41 : in order to prevent the need for paging.
0000 42
0000 43 : Since the ACP is work-queue driven, and since it is the ACP
0000 44 : that modifies the structure of the non-paged pool data base
0000 45 : including the RCB (actually a VCB) and the private structures
0000 46 : hanging off of the RCB, there is no need to obtain the system
0000 47 : data base mutex -- no races can occur. However, it is
0000 48 : necessary to raise IPL in order to stop any races with
0000 49 : NETDRIVER.
0000 50
0000 51
0000 52 : AUTHOR: A.Eldridge 14-Feb-80
0000 53
0000 54 : MODIFIED BY:
0000 55
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 datalink drivers.
0000 59 :
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 device driver.
0000 63 :
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 testing of point-to-point circuits over the NI.
0000 73 :
0000 74 :
0000 75 : V016 RNG0016 Rod Gamache 15-Apr-1983
0000 76 : Allow driver NMA parameter i.d.'s to be returned which the NETACP doesn't know about. This allows drivers the flexibility of adding parameters which the NETACP can ignore.
0000 77 :
0000 78 :
0000 79 :
0000 80 : V015 RNG0015 Rod Gamache 11-Apr-1983
0000 81 : Make the REMOVE_XXX routines call the REMOVE_DEF routine to remove the PCI or CRI CNF structure.
0000 82 :
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 to issuer of SET LINE STATE ON function.
0000 87 :
0000 88 :
0000 89 : V013 TMH0013 Tim Halvorsen 14-Feb-1983
0000 90 : Make datalink BUS parameter variable, depending on the type of datalink. It now returns a larger value for NI datalinks, to account for the additional Phase IV Ethernet route header.
0000 91 :
0000 92 :
0000 93 :
0000 94 :
0000 95 :
0000 96 :
0000 97 :
0000 98 : Conditionally allow the DMC to perform I/O functions on the controller.
0000 99 :
0000 100 : V012 TMH0012 Tim Halvorsen 20-Dec-1982
0000 101 : Add new UNA parameter to table of parameters to ignore on sensemode requests.
0000 102 : If the error qualifier returned by a datalink driver is "physical address" meaning that DECnet's requested physical address is in conflict with another user of the device, then map the qualifier to "hardware address", for which a NICE parameter exists. This is so that when an NCP user requests to have the line turned on, and it fails due to address conflict, then a parameter is returned with "bad parameter value".
0000 103 :
0000 104 :
0000 105 :
0000 106 :
0000 107 :
0000 108 :
0000 109 :
0000 110 :
0000 111 :
0000 112 :
0000 113 : V011 TMH0011 Tim Halvorsen 29-Nov-1982
0000 114 : Remove one lagging reference to NMASC_CIRTY_LAP symbol.

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

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 its 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 mnemonic 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 NFBSC ENDOFLIST, to avoid the
 0000 191 : check which doesn't allow 0 field identifiers.
 0000 192 :
 0000 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 canonical 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     $XMDDEF     : 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     $EVCFDEF     : 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     $SRCBDEF     : 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 : Macros
0000 273
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 : Define the following symbol to allow controller I/O functions on the DMC.
0000 283
0000 284 :DMC_MODEM = 0
0000 285
0000 286
0000 287 : Define special CNF flags for each database
0000 288
0000 289 :
0000 290
00000004 0000 291 PLI_V_NEW      = CNFSV_FLG_MRK1      ; New PLI being built for first time
```

```
0000 293 ; OWN STORAGE
0000 294 ; .PSECT NET_PURE,NOWRT,NOEXE,LONG
0000 295 ;
0000 296 .PSECT NET_PURE,NOWRT,NOEXE,LONG
0000 297
0000 298
0000 299 :
0000 300 : The following macros 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 309 : SWIDTH_B = 1 ; Counter width specifier for bytes
00000002 310 : SWIDTH_W = 2 ; Counter width specifier for words
00000003 311 : SWIDTH_L = 3 ; Counter width specifier for longwords
0000C000 312 NETSC_NMACNT_SLZ = <1@15>!<<$WIDTH_W>a13>!0 ; Seconds since last zeroed
0000 313
0000 314 .MACRO $COUNT_ENT base,nice,pre,mod,count,width; Insert table entry
0000 315
0000 316 .WORD <1@15>!<<$WIDTH_W>a13>!- ; Counter flag, Counter width
0000 317 <NMASC,'nice','count'> ; Nice counter i.d.
0000 318 .WORD 'pre'$"width"-mod"count" - ; Offset into internal structure
0000 319 -base ; minus internal structure base
0000 320 .ENDM $COUNT_ENT
0000 321
0000 322
0000 323 .MACRO $COUNT_TAB base,nice,pre,mod,list ; Create counter formatting table
0000 324
0000 325 .IRP A,<list>
0000 326 $COUNT_ENT base,nice,pre,mod,A ; Insert table entry
0000 327
0000 328 .ENDR
0000 329
0000 330 .LONG 0 ; Terminate the table
0000 331
0000 332 .ENDM $COUNT_TAB
0000 333
0000 334 LPD_CNT_TAB: ; Circuit counter formatting table
0000 335
0000 336 $COUNT_TAB LPDSL_ABS_TIM,CTCIR,LPD,CNT_,-
0000 337
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 350      $COUNT_TAB ....          ; Currently only "seconds since last zeroed"  
0024 351      ....                 ; is maintained outside PSIACP  
0028 352  
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  
002C 358      ....                 ; is maintained outside the driver
```

```

002C 360 ; Build a table to driver the translation between Network Management (NICE)
002C 361 ; device names to VMS devices names and to supply miscellaneous information
002C 362 ; about known devices
002C 363 ;
002C 364 ;
FFFFFFF 002C 365 NMASC_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     $$devtrn =
002C 374     .ascic "NETMAN" ; Remember where we are
002C 375     . = $$devtrn + devtrn$b_vms ; Enter Net Man device mnemonic
002C 376     .ascic "VMS" ; Advance to next field
002C 377     . = $$devtrn + devtrn$b_dev ; Enter VMS device mnemonic
002C 378     .byte devtrn$c_dev 'TYPE' ; Advance to next field
002C 379     . = $$devtrn + devtrn$b_prot ; Enter default protocol type
002C 380     .byte nmaSc_linpr 'PROT' ; Advance to device characteristics
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>, <>, <DPV>, <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>, <> : KMY-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 :
00000000 0104 425 CRI_CLR_TAB: .CNFFLD cri,l,pls    ; Polling sub-state
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 016B 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 CNVTABSC_INTRNL = ^X3FFF ; Index = INTRNL means datalink only
016C 481
016C 482 $VIELD CNVTAB,0,<
016C 483 <NMA,12,M>,- ; Define conversion table enter
016C 484 <RO,1,M>,- ; NICE protocol param i.d.
016C 485 <,3>,- ; Read-only flag
016C 486 <INT,14,M>,- ; 3 spare bits
016C 487 <FMT,2,M>,- ; Internal parameter index
016C 488 > ; Parameter format (string,value,etc)
016C 489
016C 490 .MACRO .cnvtab nma,db,param,readonly=0 ; Insert parameter i.d. table entry
016C 491
016C 492 .LONG <<nma$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$sc_int$nl @ <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 threshold
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 plci,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	.cnvtab pcli,pli,CLO	; Clock mode	
01C8	533	.cnvtab pcli,pli,PRO	; Protocol	
01CC	534	.cnvtab 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
00000000 546
00000001 0000 547 PLI_B_STATE: .BLKB 1 ; Device state
00000000 0001 548 PLI_Q_DEVNAM: .QUAD 0 ; Device name descriptor
00000000 0009 549
00000020 0009 550 DEVNAM_C_SIZ = 32 ; Size of SEARCH key buffer
00000000 0009 551 ; for line and circuit names
00000000 0009 552
00000029 0009 553 SRCH_BUF: .BLKB DEVNAM_C_SIZ ; Optional SEARCH KEY buffer
00000000 0029 554
00000000 0031 555 QIOW_Q_IOSB: .QUAD 0 ; General QIOW IOSB
00000033 0031 556 QIOW_W_RETLEN: .BLKW 1 ; Return length word for PSI QIOW
00000000 0033 557 TMP_Q_DESC: .QUAD 0 ; Scratch descriptor
00000000 003B 558
00000000 559 .PSECT NET_CODE,NOWRT,EXE
```

0000 561 .SBTTL NET\$SCAN_xxx - Scan database
0000 562 :+
0000 563 : NET\$SCAN_CRI - Scan CRI database
0000 564 : NET\$SCAN_PLI - Scan PLI database
0000 565 :
0000 566 : This co-routine is used to scan the database, and return to the caller
0000 567 : (co-routine) for each entry in the database. These routines establish
0000 568 : the order of the database entries, above that of the natural ordering of
0000 569 : the collating field.
0000 570 :
0000 571 : Inputs:
0000 572 :
0000 573 : R11 = Address of CNR
0000 574 : R10 = Address of starting CNF (or 0 if to start at the beginning)
0000 575 :
0000 576 : Outputs:
0000 577 :
0000 578 : R10 = Address of CNF if dialogue aborted prematurely, else 0.
0000 579 :
0000 580 : The caller receives control on each database entry in list (via co-routine
0000 581 : call).
0000 582 :
0000 583 : On input to co-routine:
0000 584 :
0000 585 : R0 = True if entry was found. False if at end of list (R10 invalid)
0000 586 : R10 = Address of CNF entry found
0000 587 :
0000 588 : On output from co-routine:
0000 589 :
0000 590 : R0 = CNFS_ADVANCE Advance to next CNF, continue dialogue
0000 591 : CNFS_TAKE_PREV Return previous CNF, abort dialogue
0000 592 : CNFS_TAKE_CURR Return current CNF, abort dialogue
0000 593 : CNFS_QUIT Return no CNF (R10 = 0), abort dialogue
0000 594 :
0000 595 : *** These routines must be abortable via a RET ***
0000 596 :---
0000 597 :
0000 598 NET\$SCAN_CRI::
0000 599 NET\$SCAN_PLI::
FFFD' 31 0000 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$PRE_QIO_PLI - Perform pre-QIO PLI database processing
0003 606 :: This routine is called just after validating the NFB for a database
0003 607 :: function to do any special pre-processing before the request is attempted.
0003 610 :: Here, for CRI and PLI databases, we reformat the search key into a standard
0003 611 :: format for line and circuit names, so that we don't get two database entries
0003 612 :: for DMC-0 and DMC-0.0, which should refer to the same entry.
0003 613 :: Inputs:
0003 614 :: R11 = Address of CNR
0003 615 :: NET$GQ_SRCH_KEY = Descriptor of search key value
0003 616 :: NET$GL_SRCH_ID = Field ID of search field.
0003 617 :: NET$GQ_SRCH2_KEY = Descriptor of secondary search key value
0003 618 :: NET$GL_SRCH2_ID = Field ID of secondary search field.
0003 619 :: Outputs:
0003 620 :: NET$GQ_SRCH_KEY = Reformatted search key value
0003 621 :: NET$GQ_SRCH2_KEY = Reformatted secondary search key value
0003 622 ::---
0003 623 :: 07 11 000A 624 NET$PRE_QIO_CRI::: ; Get circuit name field i.d.
0003 625 :: 000C 629 SCNFFLD cri,s,nam,R9 ; Continue in common
0003 630 :: BRB PREQIO_PRI_CRI
0003 631 :: 000C 632 NET$PRE_QIO_PLI::: ; Get line name field i.d.
0003 633 :: 000C 633 SCNFFLD pli,s,nam,R9 ; Fall thru to common code
0003 634 :: 0013 634 PREQIO_PRI_CRI:
0003 635 :: 0013 636 ; Rebuild SEARCH key if needed
0003 637 :: 0013 638 CMPL R9,NET$GL_SRCH_ID ; Is the SEARCH key the name?
0003 639 :: 00000000'EF 59 D1 0013 639 BNEQ 10$ ; If NEQ no
0003 640 :: 29 12 001A 640 MOVQ NET$GQ_SRCH_KEY,R7 ; Get the SEARCH KEY descriptor
0003 641 :: 57 00000000'EF 7D 001C 641 MOVAB SRCH BUF,R3 ; Point to the SEARCH KEY buffer
0003 642 :: 53 00000009'EF 9E 0023 642 MNEGL R3,NET$GQ_SRCH_KEY ; Prepare for size calculation
0003 643 :: 00000000'EF 53 CE 002A 643 MOVL R3,NET$GQ_SRCH_KEY+4 ; Point to new SEARCH KEY string
0003 644 :: 00000004'EF 53 DO 0031 644 BSBW REBUILD_NAME ; Rebuild the circuit name
0003 645 :: 0040 30 0038 645 BLBC R0,210$ ; If LBC then error in name
0003 646 :: 3C 50 E9 003B 646 ADDL R3,NET$GQ_SRCH_KEY ; Calculate size of the string
0003 647 :: 00000000'EF 53 CO 003E 647
0003 648 :: 0045 648
0003 649 :: 0045 649 ; Rebuild secondary SEARCH key if needed
0003 650 :: 0045 650
0003 651 10$: CMPL R9,NET$GL_SRCH2_ID ; Is the secondary SEARCH key the name?
0003 652 :: 29 12 004C 652 BNEQ 100$ ; If NEQ no
0003 653 :: 57 00000000'EF 7D 004E 653 MOVQ NET$GQ_SRCH2_KEY,R7 ; Get the secondary search key descriptor
0003 654 :: 53 00000009'EF 9E 0055 654 MOVAB SRCH BUF,R3 ; Point to the SEARCH KEY buffer
0003 655 :: 00000000'EF 53 CE 005C 655 MNEGL R3,NET$GQ_SRCH2_KEY ; Prepare for size calculation
0003 656 :: 00000004'EF 53 DO 0063 656 MOVL R3,NET$GQ_SRCH2_KEY+4 ; Point to new SEARCH KEY string
0003 657 :: 000E 30 006A 657 BSBW REBUILD_NAME ; Rebuild the circuit name
0003 658 :: OA 50 E9 006D 658 BLBC R0,210$ ; 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#SSS_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
007B 667
007B 668
007B 669
007B 670
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 overflow buf
0080 676
0080 677
0080 678
0080 679
0080 680
0080 681
0080 682
0080 683
0080 684
0080 685
0080 686
08 0A A6 91 008A 687 CMPB   DEVTRNSB_DEV(R6),#DEVTRNSC_DEV_X25 ; X.25 circuit name?
0A 0A 12 008E 688 BNEQ   1$ ; Branch if not
83 2D 90 0090 689 MOVBL  "#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$: ; Move the controller i.d. -- suppress leading zeroes. If the device
009A 693 is not a "MULTI-plexed" device then the units field may not be
009A 694 present and the name may be end with the controller specifier field
009A 695 or it may end with either the units field or the tributary field or
009A 696 both. If the device is "MULTI-plexed" then the name must contain
009A 697 at least the units field following the controller specifier field.
009A 698
009A 699
54 2D 9A 009A 700 MOVZBL "#A'-'", R4 ; Setup the field delimiter
83 54 90 009D 701 MOVB   R4,(R3)+ ; Enter it into the output buffer
00 E0 00A0 702 BBS    #DEVTRNSV_MULTI,- ; If BS then multi-unit device
14 0C A6 00A2 703 50 57 7D 00A5 704 MOVQ   R7,R0 ; Copy string descriptor
50 50 D7 00A8 705 5$: 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 NEf$BIN2ASC    ; Convert to ascii and insert
                                00C9 718
                                00C9 719
                                00C9 720
14 0C A6 00 E0 00C9 721      BBS #DEVTRNSV_MULTI,-
                                00CB 722      DEVTRNSB_CHAR(R6),50$ ; If BS then multi-unit device
                                00CE 723
                                00CE 724
                                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,TO0$    ; 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$:    :
                                00E2 735      : Device is a multi-unit device, move unit number field
                                00E2 736
83 2D 90 00E2 737      MOVB #^A'-'',(R3)+ ; Enter prefix to unit number field
54 2E 9A 00E5 738      MOVZBL #^A'.'',R4 ; Setup the field delimiter
131B 30 00E8 739      BSBW PRS_DECIMAL ; Get binary unit number
24 50 E9 00EB 740      BLBC R0,TO0$    ; 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 NEf$BIN2ASC ; Move as ascii @R3
                                00F8 745 70$:    :
                                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
0A 04 91 00F8 752      CMPB #NFBSC_DB_CRI,- ; Circuit database ?
AB 04 13 00FA 753      CNRSB_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' 00 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 $GETFLD cri,l,typ ; Get type of circuit
03 08 50 E9 012F 798 BLBC R0,5$ ; Branch if not set
03 58 D1 0132 799 CMPL R8,#NMASC_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
013A 805 5$: BSBW NET$LOCATE_LPD ; Get LPD associated with this CRI
58 18 50 E9 013D 806 BLBC R0,10$ ; If EQL then none
28 A6 9A 0140 807 MOVZBL LPDSB_PLVEC(R6),R8 ; Get PLVEC index
12 13 0144 808 BEQL 10$ ; Skip if none assigned
53 14 A6 3C 0146 809 MOVZWL LPDSW_CHAN(R6),R3 ; Get I/O channel
54 0000'8F 3C 014A 810 MOVZWL #IOS_SENSEMODE,R4 ; Get I/O function code
56 0000016C'EF 9E 014F 811 MOVAB CRI_TRN_TAB,R6 ; Get table for NICIE param translation
46 11 0156 812 BRB SHO_PLI_CRI ; Pre-process the CRI CNF block
0158 813
0158 814 10$: MOVL #1,R0 ; Report success - return parameters
0158 815 RSB
015C 816
015C 817 NET$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 PSIAACP for it's parameter values
0166 822
0166 823 $GETFLD pli,l,pro ; Get protocol parameter value

```

```

05 08 50 E9 0173 824      BLBC R0,5$          ; If not specified, skip it
      58 D1 0176 825      CMPL R8,#NMASC_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 BO 0184 833      MOVW PLVECSAW CHAN[R0],R3 ; Get I/O channel
      54 0000'8F BO 018C 834      MOVW #IOS_SENSEMODE!IO$M_CTRL,R4 ; Get QIO function code
56 000001B8'EF 9E 0191 835      MOVAB PLI_TRN_TAB,R6       ; Get table for NICe param translation
      04 11 0198 836      BRB  SHO_PLI_CRI        ; Get a list of parameters from the driver
      019A 837
      50 01 D0 019A 838 10$: MOVL #1,R0           ; Report success - return parameters
      05 019D 839      RSB
      019E 840
      019E 841 SHO_PLI_CRI:
      019E 842
      019E 843
      019E 844      ; Init IOSB image
      00000029'EF 7C 019E 845      CLRQ QIOW_Q_IOSB
      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 assume unsupported by driver
      13 00000029'EF E9 01CA 853      BLBC QIOW_Q_IOSB,100$ ; If LBC assume unsupported by driver
      0B 0000002D'EF 0B 01D1 854      BBC #XMSV_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$: MOVL #1,R0           ; Report success
      05 01E7 859 110$: RSB            ; Return status in R0

```

```

01E8 861      .SBTTL JAM_CNF - Store driver values into CNF
01E8 862      + JAM_CNF - Store datalink driver parameters into CNF
01E8 863      This routine updates the CNF with the latest and greatest parameter
01E8 864      values from the datalink driver.
01E8 865      Inputs:
01E8 866      01E8 867      R11 = Address of CNR
01E8 868      01E8 869      R10 = Address of CNF
01E8 870      01E8 871      R6 = Address of datalink-NICE translation table
01E8 872      01E8 873      R4 = Address of buffer returned by SENSEMODE
01E8 873      01E8 874      QIOW_Q_IOSB = IOSB from SENSEMODE operation
01E8 874      01E8 875      Outputs:
01E8 875      01E8 876      R0 = Status code
01E8 876      01E8 877      .ENABL LSB
01E8 877      01E8 878      00000000'EF DD 01E8 882 JAM_CNF:
01E8 878      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
58 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      :
57 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			
		0238	920			
		0242	921	70\$:	TSTW	QIOW_Q_IOSB+2
00000002B'EF	OB 59 0C	E5	0238	921	BBS	#12,R9,80\$
		E0	023E	922		; Deplete bytes
			0242	923		; Br if string value
			0242	924		
			0242	925		
57 04	D0	0242	926		MOVL	#4,R7
2C	10	0245	927		BSBB	TAKE_R7_IOSB
54 04	C0	0247	928		ADDL	#4,R4
FFA9	31	024A	929		BRW	10\$
			024D	930		
			024D	931		
			024D	932		
57 21	10	024D	933	80\$:	BSBB	TAKE_2_IOSB
84 32	024F	934			CVTWL	(R4)+,R7
15 19	0252	935			BLSS	120\$
54 57	C0	0254	936		ADDL	R7,R4
1A 10	0257	937			BSBB	TAKE_R7_IOSB
FF9A	31	0259	938		BRW	10\$
			025C	939		
00000000'EF	8ED0	025C	940	100\$:	POPL	NET\$GL_FLAGS
	05	0263	941		RSB	
		0264	942			; Return status in R0
50 01	D0	0264	943	110\$:	MOVL	#1,R0
F3	11	0267	944		BRB	100\$
		0269	945			; Return success
50 0000'8F	3C	0269	946	120\$:	MOVZWL	#SSS_RESULTOVF,R0
EC	11	026E	947		BRB	100\$
		0270	948			; Indicate buffer error
57 02	3C	0270	949			
0000002B'EF	57 A2	0273	950		MOVZWL	#2,R7
04 18	027A	951				; Setup bytes to take
8E D5	027C	952			SUBW	R7,QIOW_Q_IOSB+2
E9 11	027E	953			BGEQ	200\$
05 0280	0281	954			TSTL	(SP)+
		955			BRB	120\$
		956	200\$:		RSB	
		957				
		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 $GETFLD cri,l,use : Get USAGE parameter
0281 985 BLBC R0,5$ : Branch if not set - defaulting error
0281 986 CMPL R8,#NMASC_CIRUS_PER : PVC?
0281 987 BEQL 10$ : If not, then exit successfully
0281 988 5$: MOVL #1,R0 : Do nothing
0281 989 RSB :
029A 990 :
029A 991 : Issue a SHOW function to PSI for its circuit parameters
029A 992 :
029A 993 10$: $CNFFLD cri,s,nam,R9 : Set field ID of circuit name
029A 994 BSBW SEND TO_PSI : Call co-routine to setup NFB, etc.
029A 995 BLBC R0,90$ : Branch if error detected
029A 996 MOVBL #NFBSC_FC_SHOW_NFB$B_FCT(R6) : Set function code
029A 997 PUSHR #^M<R2,R3,R4,R5> : Save registers
029A 998 ASSUME NFBSC_ENDOFLIST EQ 0 : NFB terminator and PSI_TAB are same
029A 999 MOVC #CRI_PSI_SIZ,CRI_PSI_TAB,(R2) : Copy in the field IDs
029A 1000 POPR #^M<R2,R3,R4,R5> : Restore registers
029A 1001 MOVAB CRI_PSI_SIZ(R2),R2 : Increment field ID list pointer
029A 1002 ADDL #P4BUF_SIZE,R3 : Set size of P4 return buffer
029A 1003 JSB a(SP)+ : Call SEND_TO_PSI back to issue QIO
029A 1004 BLBS R0,STORE_PSI_PARAMS : Branch if ok
029A 1005 90$: RSB : Return with error
029A 1006 :
029A 1007 GET_PSI_PLI:
029A 1008 :
029A 1009 : Issue a SHOW function to PSI for its line parameters
029A 1010 :
029A 1011 $CNFFLD pli,s,nam,R9 : Set field ID of circuit name
029A 1012 BSBW SEND TO_PSI : Call co-routine to setup NFB, etc.
029A 1013 BLBC R0,90$ : Branch if error detected
029A 1014 MOVBL #NFBSC_FC_SHOW_NFB$B_FCT(R6) : Set function code
029A 1015 PUSHR #^M<R2,R3,R4,R5> : Save registers
029A 1016 ASSUME NFBSC_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	
53	00000200'8F	52 38 A2	9E	02E3	1019		MOVAB	PLI_PSI_SIZ(R2),R2	Increment field ID list pointer	
		9E	C0	02E7	1020		ADDL	#P4BUF_SIZE,R3	Set size of P4 return buffer	
		16	02EE	1021			JSB	@(SP)+	Call SEND_TO_PSI back to issue QIO	
		01 50	E8	02F0	1022	90\$:	BLBS	R0,STORE_PSI_PARAMS	Branch if ok	
		05	02F3	1023			RSB		Return with error	
			02F4	1024						
			02F4	1025	STORE_PSI_PARAMS:					
			02F4	1026						
			02F4	1027						
			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,#NFBSS_TYP,R9	#NFB\$C_TYP_STR ; Is it a string?	
		OC	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	#SSS_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 - Clear 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 in CNF, ignore errors
F6 11 0368 1086 BRB 10\$; Loop
05 036A 1087 20\$: RSB

```

036B 1089 .SBTTL NET$DEFAULT_CRI - Apply default values
036B 1090 :+
036B 1091 : NET$DEFAULT_CRI - Apply default values to selected CNF parameters.
036B 1092 :
036B 1093 : This routine is called by CNFS$INSERT just prior to validating a CNF
036B 1094 : entry which is to be inserted into the database. Its purpose is to
036B 1095 : supply default values to selected parameters.
036B 1096 :
036B 1097 : INPUTS: R11 CNR pointer
036B 1098 : R10 CNF pointer
036B 1099 :
036B 1100 : OUTPUTS: R11 CNR pointer
036B 1101 : R10 CNF pointer
036B 1102 : R0 Status code.
036B 1103 :
036B 1104 :
036B 1105 : All other registers contain garbage.
036B 1106 NET$DEFAULT_CRI:
FC92' 30 036B 1107 BSBW NET$APPLY_DFLT ; Apply standard table defaults
103F 30 036E 1108 $GETFLD cri,s,nam ; Get descriptor of circuit name
75 50 E9 037B 1109 BSBW PRS_MNEMONIC ; Locate the table entry
037E 1110 BLBC R0,T0$ ; If not found, assume DDC
0381 1111 :
0381 1112 :
0381 1113 :
0381 1114 : If the circuit type is not specified yet, and the name
0381 1115 : the form X25-x, then apply the default X25 circuit parameters
(including setting it's type to "X25").
0381 1116 :
0381 1117 :
0381 1118 :
0381 1119 :
0381 1120 :
0381 1121 :
0381 1122 :
0381 1123 :
0381 1124 :
0381 1125 :
0391 1126 :
0391 1127 :
03A1 1128 :
03A1 1129 :
03A1 1130 :
03A9 1131 :
03AB 1132 4$: BRB 5$ ; If X.25 datalink mapping is enabled, then apply the Transport
03AE 1133 : MOVL #1,R8 ; defaults.
03BB 1134 :
03BB 1135 :
03BB 1136 :
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 $GETFLD cri,l,use ; Get circuit usage
02 58 D1 03E5 1144 CMPL R8,#NMASC_CIRUS_OUT ; Outgoing circuit?
28 12 03E8 1145 BNEQ 20$ ; Skip if not

```

56 00000000'EF, 9E 03EA 1146 MOVAB NET\$G_CRI_DLMOUT,R6 ; Get address of outgoing DLM defaults
FCOC' 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 DEVTRNSB_DEV(R6),#DEVTRNSC_DEV_UNA ; Is it an NI circuit?
0 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
OA 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

```

        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 $GETFLD pli,s,nam : Get descriptor of line name
OF8A 30 0430 1190 BSBW PRS MNEMONIC : Locate the table entry
01 50 E8 0433 1191 BLBS R0,T0$ : 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 mnemonic.
        0437 1196 :
        0437 1197 $GETFLD pli,l,pro : Was protocol explicitly specified?
        0444 1198 BLBS R0,20$ : If so, don't change it
        0447 1199 MOVZBL DEVTRNSB PROT(R6),R8 : Get default protocol type
FF 8F 58 9A 044B 1200 CMPB R8 #NMASC_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 DEVTRNSB DEV(R6),- : Is this a "point-to-point" XE?
        0A 0457 1208 #DEVTRNSC_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 $DISPATCH R8,<- : Based on line protocol,
        046A 1213 <NMASC_LINPR_POI,30$>- : DDCMP Point-to-point
        046A 1214 <NMASC_LINPR_CON,30$>- : DDCMP Controller station
        046A 1215 <NMASC_LINPR_TRI,30$>- : DDCMP Tributary
        046A 1216 <NMASC_LINPR_DMC,30$>- : DDCMP DMC compatible mode
        046A 1217 <NMASC_LINPR_LAPB,40$>- : X.25 LAPB protocol
        046A 1218 <NMASC_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
57 7E 7C 04A0 1244 CLRQ -(SP) ; Zero 8 byte buffer
06 D0 04A2 1245 MOVL #6,R7 ; Set length of string to 6 bytes
58 SE 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,RO ; 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$INSERT_CRI::
04BC 1269          CLRL -(SP)           ; Scratch space on stack for STATE
50   0000'8F  D4 04BE 1270          MOVZWL #SS$ INSFARG,R0    ; Set error if parameter not present
03   50   E8 04D0 1271          $GETFLD cri_l,sta      ; Get the operator state
009A 31 04D3 1272          BLBS R0,5$           ; Error if not already defaulted
6E   58  D0 04D6 1273          BRW 200$            ; Exit with status
04D9 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          $GETFLD cri_v,ser      ; See if service functions are enabled
50   OF 50   E9 04E6 1279          BLBC R0,10$          ; Branch if parameter not set
0C   58   E9 04E9 1280          BLBC R8,10$          ; If LBC then "service" state is legal
02   6E   D1 04F3 1281          MOVL #SS$ BADPARAM,R0 ; Assume state is "service"
78   13 04F6 1282          CMPL (SP),#NMASC_STATE_SER ; Is it?
04F8 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          ;
050A 1292          ; If the circuit type associated with this CNF is X25, then ship
050A 1293          ; the X.25 specific circuit parameters to PSIACP via a control QIO.
050A 1294          ; If the QIO succeeds, then allow the CNF to be inserted. In
050A 1295          ; either case, we do not perform any datalink processing.
050A 1296          $GETFLD cri_l,typ      ; Get circuit type
03 50  E9 0517 1297          BLBC R0,25$          ; If error, assume not X.25
03 58  D1 051A 1298          CMPL R8,#NMASC_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,#NMASC_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

```

5A	8ED0	05	0543	1309	RSB				: Return - do not allow insertion
18	11	0544	1310	26\$:	POPL	R10			: Restore current CNF
		0547	1311		BRB	50\$; and allow Transport to use it
		0549	1312						
22	28	50	10	0549	1313	28\$:	BSBB	CRI TO PSI	: Tell it to PSIACP
		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				
OC	50	09	58	E9	054E	1321	29\$:	\$GETFLD cri,v,DLM	: X.25 datalink mapping enabled?
					055B	1322		BLBC R0,60\$: If not set, do not use this circuit
					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	6E	D0	FA99'	30	0561	1328	50\$:	MOVL (SP),R8	: Get STATE value
					0564	1329		RSBW NET\$DLL_OPR_SET	: Signal operator event
06	50	E9	06	50	0567	1330		BLBC R0,200\$: If LBC then error
					056A	1331			
					056A	1332			: Start counter timer ticking
					056A	1333			
50	FA93'	30	01	D0	056A	1334	60\$:	BSBW NET\$SET_CTR_TIMER	: Reset automatic counter timer
					056D	1335		MOVL #1,RO	: Indicate success
			8E	D5	0570	1336	200\$:	TSTL (SP)+	: Cleanup the stack
				05	0572	1337		RSB	: Return status in R0

```

      0573 1339 .SBTTL CRI_TO_PSI - Send CRI parameters to PSIACP
      0573 1340 :+ CRI_TO_PSI - Issue SET of CRI parameters to PSIACP
      0573 1341 : Send the CRI parameters that PSIACP cares about via a ACP control QIO.
      0573 1342 :
      0573 1343 :
      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 CRI_TO_PSI:
      0573 1355 :
      0573 1356 : Check if required parameters are present
      0573 1357 :
      0573 1358 50 0000'8F 3C 0573 1359 MOVZWL #SS$_INSFARG,R0 ; Set error if parameter not present
      66 50 E9 0578 1360 $GETFLD cri[_use ; Check if USAGE parameter specified
      0585 1361 BLBC R0,90$ ; If not, error - must be specified
      0588 1362 :
      0588 1363 :
      0588 1364 : If this is an incoming SVC, then make sure that NETACP is
      0588 1365 : declared (to PSI) as being able to accept incoming calls
      0588 1366 : since PSI might not have been up earlier to receive the
      0588 1367 : declaration when the SUBADDRESS parameter was set.
      01 58 D1 0588 1368 CMPL R8,#NMASC_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
      FASE' 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$: :
      05A9 1377 :
      05A9 1378 : If this is a PVC, then go ahead and tell PSI. If it's a
      05A9 1379 : incoming or outgoing SVC, then don't bother, since PSI
      05A9 1380 : doesn't need to know about them.
      00 58 D1 05A9 1381 CMPL R8,#NMASC_CIRUS_PER ; PVC?
      40 12 05AC 1382 BNEQ 90$ ; If not, then exit successfully
      05AE 1383 :
      05AE 1384 :
      05AE 1385 :
      05AE 1386 :
      0383 30 05B5 1387 $CNFFLD cri,s,nam,R9 ; Set field ID of circuit name
      33 50 E9 05B8 1388 BSBW SEND_TO_PSI ; Call co-routine to setup NFB, etc.
      66 23 90 05BB 1389 BLBC R0,90$ ; Branch if error detected
      59 85 D0 05BE 1390 MOVAB #NFB$C_FC_SET,NFB$B_FCT(R6) ; Store function code
      22 13 05C5 1391 10$: MOVAB CRI_PSI_TAB,R5 ; Point to PSI circuit parameters
      FA33' 30 05CA 1392 MOVL (R5)+,R9 ; Get next parameter ID
      F5 50 E9 05CD 1393 BEQL 50$ ; Branch if end of table
      82 59 D0 05D0 1394 BSBW CNF$GET_FIELD ; Get the parameter value
      MOVL R9,(R2)+ ; Skip it if it doesn't exist
      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 #NFBSC_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
 it into the database.
 05EF 1414 :
 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 :
 00000000'EF 58 90 05FC 1430 \$GETFLD pli,l,sta ; Get the operator state
 0603 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 :
 05 12 50 E9 0610 1438 \$GETFLD pli,l,pro ; Get protocol type
 05 58 D1 0613 1439 BLBC R0,25\$; If error, assume not X.25
 0D 12 0616 1440 CMPL R8,#NMASC_LINPR_LAPB ; LAPB protocol?
 02D6 30 0618 1441 BNEQ 25\$; Branch if not
 06 50 E9 061B 1442 BSBW PLI_TO_PSI ; If so, send it to PSIACP
 F9DF' 30 061E 1443 BLBC R0,24\$; Branch if error detected
 50 01 DD 0621 1444 BSBW NET\$SET_CTR_TIMER ; Start/reset automatic counter timer
 05 0624 1445 MOVL #1,R0 ; Set successful
 5A 56 DD 0625 1446 24\$: RSB ; Exit with status
 1E 13 062A 1447 25\$: PUSHL R10 ; Save pointer to current CNF
 0E 50 E9 0639 1450 MOVL R6,R10 ; Point to old version of CNF
 05 58 D1 063C 1451 BEQL 26\$; Branch if none
 09 12 063F 1452 \$GETFLD pli,l,pro ; Get old protocol type
 50 0000'8F 3C 0641 1453 BLBC R0,26\$; If error, assume still not X.25
 SA 8ED0 0646 1454 CMPL R8,#NMASC_LINPR_LAPB ; Did it used to be LAPB?
 05 0649 1455 BNEQ 26\$; Branch if not
 SA 8ED0 064A 1456 MOVZWL #SSS_WRITLCK,R0 ; Do not allow protocol change w/o CLEAR
 064D 1457 26\$: POPL R10 ; Restore current CNF
 064D 1458 RSB ; Return - do not allow insertion
 064D 1459 POPL R10 ; Restore current CNF
 064D 1460 : Locate the PLVEC index for this PLI. If there is none then
 064D 1461 : assign and initialize a PLVEC entry for this PLI.
 54 12 AA 3C 064D 1462 MOVZWL CNFSW_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

```

01 00000000'EF44    90 0659 1466      SETBIT PLI_V_NEW,CNF$B_FLG(R10); Indicate PLVEC has been newly assigned
                                MOVB #NMASC_STATE_OFF,- ; Init to "off" state
                                065D 1467
                                065F 1468      PLVEC$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 50 00000000'EF  D0 066F 1476      MOVL NET$GL_PTR_VCB,R0 ; Get RCB address
                                008A C0 9A 0676 1477      MOVZBL RCB$BETY(R0),R0 ; Get executor type
                                067B 1478      $DISPATCH R0,2- ; Allow only if set to one of:
                                067B 1479      <ADJ$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 #SSS_BADPARAM,R0 ; Else, illegal protocol
                                068A 1483      $CNFFLD pli,[.pro,R9] ; Indicate parameter in error
                                5B 11 0691 1484 29$:      BRB 100$ ; Exit with error
                                0693 1485 28$:      ; Issue the line control QIO to setup the driver with the control
                                0693 1486      information. The IOS_STARTUP and IOS_SHUTDOWN modifiers are
                                0693 1487      illegal unless there is a state change.
                                0693 1488
                                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      -P[VECSAB_STATE[R4]]
                                05 12 06A9 1494      BNEQ 40$ ; If NEQ then state change
                                0000'8F B0 06AB 1495      MOVW #IOSM_STARTUP!- ; Specify unwanted I/O modifiers
                                51 06AF 1496      IOSM_SHUTDOWN,R1
                                00000000'EF 90 06B0 1497 40$:      MOVB PLI_B_STATE,- ; Setup new state value
                                00000000'EF44 06B6 1498      -P[VECSAB_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$SET_QIOW ; Issue set characteristics QIO
                                59 51 06CE 1502      MOVL R1,R9 ; Return 2nd IOSB longword as well
                                00000000'EF44 8E F6 06D1 1503      CVTLB (SP)+,PLVEC$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
                                06E8 1507      ; Start counter timer ticking
                                06E8 1508
                                06E8 1509
                                50 F915' 30 06E8 1510      BSBW NET$SET_CTR_TIMER ; Reset automatic counter timer
                                01 D0 06EB 1511      MOVL #1,R0 ; Set successful
                                06EE 1512
                                06EE 1513      ; If the PLI is not new then simply return the status in R0. If
                                06EE 1514      the PLI is new then the newly allocated PLVEC slot must be
                                06EE 1515      deallocated if R0 indicates an error.
                                06EE 1516
                                08 08 04 E5 06EE 1517 100$:      BBCC #PLI_V_NEW,- ; Br unless PLVEC has been newly
                                AA 06F0 1518      CNF$B_FLG(R10),110$ ; assigned
                                08 50 E8 06F3 1519      BLBS R0,110$ ; If B5 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

```

- Datalink database action routines ^{C 2} 16-SEP-1984 01:15:07 VAX/VMS Macro V04-00
NET\$INSERT_PLI - Pre-insertion processin 5-SEP-1984 02:18:18 [NETACP.SRC]NETCNFDLL.MAR;1 Page 35
(17)

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
      54 50 0000'8F 3C 06FF 1540 MOVZWL #SS$ INSFMEM,R0 : Assume no resources for this entry
      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$AB_REF[R4] : Is this cell free?
      06 13 0712 1543 BEQL 20$ : 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$AB_REF[R4] : Claim the PLVEC cell
      00000000'EF44 B4 0725 1548 CLRW PLVEC$AW_CHAN[R4] : No channel assigned yet
      50 0000'8F 3C 072C 1549 MOVZWL #SS$_NOSUCHDEV,R0 : Assume error
      D6 50 E9 073E 1550 $GETFLD pli$nam : Else get the line name
      0731 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 :
      53 5E 17 C2 0741 1556 SUBL #MAX_C_DEVNAM+8,SP : Create scratch space on stack
      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
      OC1A 30 074F 1560 BSBW TRAN_DEVNAM : Translate device name to VMS format
      0752 1561 BLBC R0,40$ : - advances R3, table entry -> R6
      50 0000'8F 3C 0755 1562 MOVZWL #SS$_IVDEVNAM,R0 : If LBC then error
      57 D5 075A 1563 TSTL R7 : Assume error
      25 12 075C 1564 BNEQ 40$ : Any unparsed characters?
      54 12 AA 3C 075E 1565 MOVZWL CNF$W_ID(R10),R4 : If NEQ yes, name format error
      0A A6 90 0762 1566 MOVB DEVTRNSB_DEV(R6) - : Recover R4
      00000000'EF44 0765 1568 PLVEC$AB_DEV[R4] : Enter device type code
      6E 53 C0 0768 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
      00000000'EF44 3C 0789 1576 MOVZWL PLVEC$AW_CHAN[R4],R0 : Get I/O channel
      00000000'GF 16 0791 1577 JSB G^IOCSVER,FYCHAN : Return CCB in R1 regardless of error
      61 D0 0797 1578 MOVL CCB$L_UCB(R1),- : Store device UCB
      00000000'EF44 0799 1579 PLVEC$AL_UCB[R4] : 
      5E 04 C2 079F 1581 SUBL #4,SP : Use stack for device characteristics
  
```

53 5E DD 07A2 1582 MOVL SP,R3 : Point R3 at return buffer
 00000031'EF 00 DD 07A5 1583 PUSHL #0 : Terminate GETDVI item list
 53 53 DD 07A7 1584 PUSHAB QIOW_W_RETLEN : Point to result length word
 7E 02 B0 07AD 1585 PUSHL R3 : Point to DEVCCHAR return buffer
 7E 04 B0 07AF 1586 MOVW #DVIS_DEVCHAR,-(SP) : Item code
 51 5E DD 07B2 1587 MOVW #4,-(SP) : Size of DEVCCHAR result buffer
 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 :
 50 55 50 E9 07D6 1594 49\$: BLBC R0,120\$: Branch if error detected
 0000'8F 3C 07D9 1595 MOVZWL #\$\$\$ IVDEVNAM,R0 : Assume error
 4C 63 OD 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
 55 58 0000004C 8F E9 07EF 1605 BLBC R0,70\$: If not set, skip it
 C1 07F2 1606 ADDL3 #CXB\$C OVERHEAD,R8,R5 : Add in datalink's overhead
 07FA 1607 \$GETFLD pli,l,5fn : 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 DD 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 DD 082A 1619 70\$: MOVL #1,R0 : Success
 05 082D 1620 100\$: RSB : Return status in R0
 082E 1621 :
 082E 1622 120\$: : DEASSIGN the channel on error
 082E 1623 :
 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$GET_FIELD ; Get datalink DLLQIO buffer
      7D 5v E9 084C 1654 BLBC R0,100$ ; If LBC then error
      68 51 CA 084F 1655 BICL R1,DLLQIOSL_FUNC(R8) ; Mask out illegal I/O modifier bits
      04 A8 D5 0852 1656 TSTL DLLQIOSL_P1(R8) ; Any P1 buffer ?
      04 13 0855 1657 BEQL 5$ ; If EQL no
      58 C0 0857 1658 ADDL R8,DLLQIOSL_P1(R8) ; Else turn offset into a pointer
      8A8 D5 085B 1659 5$: TSTL DLLQIOSL_P2(R8) ; Any P2 buffer ?
      0C 13 085E 1660 BEQL 10$ ; If EQL then none
      58 C0 0860 1661 ADDL R8,DLLQIOSL_P2(R8) ; Turn offset into a pointer
      08 A8 D0 0864 1662 MOVL DLLQIOSL_P2(R8),R0 ; Point to P2 buffer descriptor
      58 C0 0868 1663 ADDL R8,4(R0) ; Turn offset into a pointer
      086C 1664 10$: $QIOW_S - ; Set the device characteristics
      086C 1665 EFN = #NET$C_EFN_WAIT,-
      086C 1666 !OSB = QIOW_Q_IOSB,-
      086C 1667 CHAN = R2,-
      086C 1668 FUNC = DLLQIOSL_FUNC(R8),-
      086C 1669 P1 = @DLLQIOSL_P1(R8),- ; This is a PUSHAL
      086C 1670 P2 = DLLQIOSL_P2(R8),-
      086C 1671 P3 = DLLQIOSL_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$: MOVL QIOW_Q_IOSB+4,R9 ; Get remainder of IOSB
      00000B04 8F 59 D1 08A1 1675 CMPL R9,#NMASC_PCLI_PHA ; "physical address"?
      07 12 08A8 1676 BNEQ 22$ ; If so,
      59 00000488 8F D0 08AA 1677 MOVL #NMASC_PCLI_HWA,R9 ; then map it to "hardware address"
      0A70 30 08B1 1678 ; (so that we see it at the NICE level)
      08B4 1679 22$: BSBW NET$CVT_NMA_INT ; Convert NMA parameter i.d. to an
      3FFF 8F 59 B1 08B4 1681 CMPW R9,#CNVTABSC_INTRNL ; internal parameter i.d.
      07 12 08B9 1682 BNEQ 25$ ; Datalink only parameter?
      59 0000002D'EF D0 08BB 1683 MOVL QIOW_Q_IOSB+4,R9 ; Branch if not
      50 00000029'EF 7D 08C2 1684 25$: MOVQ QIOW_Q_IOSB,R0 ; If so, I'd rather see the NMA code
      08C9 1685 ; Get I/O status
      08C9 1686 ; Log the final status into the journal file, if any
      08C9 1687 ;

```

00000029'EF	58	68	3C	08C9	1688		MOVZWL	DLLQIO\$L FUNC(R8),R8	: Pass I/O function code into journal
	50	7D	08CC	1689	100\$:		MOVQ	R0_QIOW @ IOSB	: Save final status
	F72A'	30	08D3	1690			BSBW	#ET\$JNX-C0	: 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	81 58	B0	08DD	1693			MOVW	R8_(R1)+	: I/O function code
	0000029'EF	7D	08E0	1694			MOVQ	QIOW_Q_IOSB,(R1)+	: I/O status quadword
50	9E 16	08E7	1695				JSB	@(SP)+-	: Log the journalling record
	0000029'EF	7D	08E9	1696	120\$:		MOVQ	QIOW_Q_IOSB,R0	: Restore final status
	05	08F0	1697				RSB		

```

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
      37 10 0902 1718 $CNFFLD plis,nam,R9    ; Set field ID of line name
      33 50 E9 0904 1720 BSBB  SEND_TO_PSI     ; Call co-routine to setup NFB, etc.
      66 23 90 0907 1721 BLBC  R0,90$        ; Branch if error detected
      59 85 D0 0911 1722 10$: MOVAB  #NFB$C_FC_SET,NFB$B_FCT(R6) ; Store function code
      22 13 0914 1723 MOVL  (R5)+,R9       ; Point to PSI Line parameters
      F6E7' 30 0916 1725 BSBW  CNF$GET_FIELD  ; Get next parameter ID
      F5 50 E9 0919 1726 BLBC  R0,10$        ; Branch if end of table
      82 59 D0 091C 1727 MOVL  R9,(R2)+      ; Get the parameter value
      02 10 ED 091F 1728 CMPZV #NFB$V_TYP,#NFB$S_TYP,- ; Skip it if it doesn't exist
      02 59 0922 1729 R9,#NFB$C_TYP_STR    ; Store into NFB
      05 13 0924 1730 BEQL  20$             ; String or binary?
      83 58 D0 0926 1731 MOVL  R8,(R3)+      ; Branch if string
      E6 11 0929 1732 BRB   10$             ; Store longword into P4 buffer
      83 57 B0 092B 1733 20$: MOVW  R7,(R3)+      ; Store string count word
      34 BB 092E 1734 PUSHR #^M<R2,R4,R5> ; Save registers
      63 68 57 28 0930 1735 MOVC  R7,(R8),(R3) ; Store string text
      34 BA 0934 1736 POPR  #^M<R2,R4,R5> ; Restore registers
      D9 11 0936 1737 BRB   10$             ; Call SEND_TO_PSI back to issue QIO
      9E 16 0938 1738 50$: JSB   @(SP)+      ; Return with status
      05 093A 1739 90$: RSB

```

```

093B 1741      .SBTTL SEND_TO_PSI - Send control QIO to PSIACP
093B 1742 :+
093B 1743 : SEND_TO_PSI - Send ACP control QIO to PSIACP
093B 1744 :
093B 1745 : This routine is called to transmit an ACP control function to PSIACP
093B 1746 : on behalf on a local CNF, such as a LAPB line. CNFs like these are
093B 1747 : "known" about by both NEACP and PSIACP, NETACP in order to perform
093B 1748 : network management functions, and PSIACP in order to manage the line.
093B 1749 :
093B 1750 : Inputs:
093B 1751 :
093B 1752 :   R11 = Address of CNR
093B 1753 :   R10 = Address of CNF
093B 1754 :   R9 = Address of parameter ID describing entity name
093B 1755 :
093B 1756 : Outputs:
093B 1757 :
093B 1758 :   R0 = Status code
093B 1759 :
093B 1760 : If the function code is NFB$C_FC_SHOW or NFB$C_FC_ZERCOU, then:
093B 1761 :
093B 1762 :   R7/R8 = Descriptor of result P4 buffer
093B 1763 :   R9 = Address of scratch storage, which should be deallocated
093B 1764 :           when processing of the P4 buffer is completed.
093B 1765 :
093B 1766 :   R1-R6 are destroyed.
093B 1767 :
093B 1768 : This routine makes a co-routine call back to the caller after setting
093B 1769 : up the NFB and associated control QIO buffers, so that the caller can
093B 1770 : fill in the essential fields. When the caller returns via co-routine
093B 1771 : call to this routine, the control QIO is issued to PSIACP.
093B 1772 :
093B 1773 : On input to co-routine:
093B 1774 :
093B 1775 :   R6 = Address of NFB
093B 1776 :   R3 = Address of P4 buffer
093B 1777 :   R2 = Address of NFB$L_FLDID list
093B 1778 :
093B 1779 : On return from co-routine:
093B 1780 :
093B 1781 :   R3 = Updated pointer to end of P4 buffer
093B 1782 :   R2 = Updated pointer to end of FLDID list
093B 1783 :
093B 1784 :   R0-R1,R4-R5,R7-R9 may be destroyed by co-routine.
093B 1785 :-
093B 1786 :
093B 1787 NFB_SIZE = NFB$C_LENGTH + 30*4 ; allow for all PSI field IDs
093B 1788 P2BUF_SIZE = 4+2*DEVNAM_C_SIZ+NFB$C_CTX_SIZE ; P2 holds CNF count and entity ID
093B 1789 : plus context area
093B 1790 P4BUF_SIZE = 512 ; P4 holds all parameter values
093B 1791 :
093B 1792 ASSUME PLI_PSI_SIZ LE 30*4 ; Must be enough room for PLI params
093B 1793 :
093B 1794 SEND_TO_PSI:
093B 1795   MOVZWL #12+NFB_SIZE+P2BUF_SIZE+P4BUF_SIZE,R1 ; Set size of block overhead
0940 1796 : plus NFB, P2 and P4 buffer sizes
0940 1797   JSB     NET$ALLOCATE ; Allocate the storage

```

51 032A 8F 3C

00000000'EF 16 0940 1797

		01 50	E8 0946	1798	BLBS	R0,5\$: Branch if successful
		52	05 0949	1799	RSB		: Else, exit with error status
66	10	00 56 0C A2	DD 094A	1800	5\$: PUSHL	R2	: Save address of block
		02 A6 0E 00	9E 094C	1801	MOVAB	12(R2),R6	: Point to scratch NFB
		01 A6 04	2C 0950	1802	MOVC5	#0,(SP),#0,#NFBSC_LENGTH,(R6)	: Zero it
		03 A6 00	90 0956	1803	MOVB	CNRSB_TYPE(R11),NFB\$B_DATABASE(R6)	: Store database ID
		04 A6 59	90 095B	1804	MOVB	#NFB\$M_NOCTX,NFB\$B_FLAGS(R6)	: Indicate one-shot request
		F696'	90 095F	1805	MOVB	#NFBSC_OP_EQL,NFB\$B_OPER(R6)	: Search for key EQL
		03 50	D0 0963	1806	MOVL	R9,NFB\$L_SRCH_KEY(R6)	: Search for specific entry
		00BC C6 57	30 0967	1807			: by it's network management name
00BE		00E8 C6 68	E8 096A	1808	BSBW	CNF\$GET_FIELD	: Get descriptor of name
		83	31 096D	1809	BLBS	R0,10\$: Branch if ok
		52 10 A6	B0 0970	1811	10\$: BRW	110\$: Abort if unable to get entity name
		53 011E C6	28 0975	1812	MOVW	R7,NFB_SIZE+4(R6)	: Store size of string
		50 01	B4 097B	1813	MOVC	R7,(R8),NFB_SIZE+6(R6)	: Copy into buffer
		9E 16	9E 097D	1814	CLRW	(R3)+	: Make context a null string
		09A3	9E 0981	1815	MOVAB	NFB\$L_FLDID(R6),R2	: Point to NFB field ID list
		09A3	D0 0986	1816	MOVAB	NFB_SIZE+P2BUF_SIZE(R6),R3	: Get address of P4 buffer
		09A3		1817	MOVL	#1,R0	: Indicate co-routine init. successful
		09A3		1818			: Call the caller back so that he can fill in the NFB and P4 buffers.
		09A3		1819			: Swap block pointer & return address
		09A3		1820	SWAP	(SP),4(SP)	: so that caller's address is on top
		09A3		1821	JSB	a(SP)+	: Fill in the NFB and P4 buffers
		09A3		1822	SWAP	(SP),4(SP)	: Swap return address & block pointer
		09A3		1823			: Terminate the NFB field ID list
		09A3		1824			: Construct descriptors of NFB, P2 and P4 buffers for QIO
		82	D4 09A3	1825	ASSUME	NFBSC_ENDOFLIST EQ 0	
		09A5		1826	CLRL	(R2)+	: Terminate NFB parameter ID list
		09A5		1827			: Construct descriptors of NFB, P2 and P4 buffers for QIO
		09A5		1828			: Construct descriptors of NFB, P2 and P4 buffers for QIO
7E	04 AE	7E 6E 0C	C1 09A5	1829			: Construct descriptors of NFB, P2 and P4 buffers for QIO
		7E 52 6E	C3 09A9	1830	ADDL3	#12,(SP),-(SP)	: Retrive starting address of NFB
		0000011E 8F	C1 09AD	1831	SUBL3	(SP),R2,-(SP)	: Compute length of NFB
51	0C AE	7E 53 6E	C3 09B6	1832	ADDL3	#NFB_SIZE+P2BUF_SIZE,4(SP),-(SP)	: Construct descriptor of P4
		000000B8 8F	C1 09BA	1833	SUBL3	(SP),R3,-(SP)	: Compute length of P4 buffer
		50 04 A1	3C 09C3	1834	ADDL3	#NFB_SIZE,12(SP),R1	: Get address of P2 buffer
		00000046 8F	C0 09C7	1835	MOVZWL	4(R1),R0	: Pick up length of entity name
		09CE	09CE	1836	ADDL	#4+2+NFBSC_CTX_SIZE,R0	: Add 4 bytes for CNF count
		09CE	09CE	1837			: plus 2 bytes for entity string size
		09CE	09CE	1838			: and context area size
		7E 50	7D 09CE	1839	MOVQ	R0,-(SP)	: Push P2 descriptor onto stack
		52 5E	D0 09D1	1840	MOVL	SP,R2	: Point to P2/P4/NFB descriptors
53	08 A2	9E 09D4	1841	MOVAB	8(R2),R3	: Point to P4 descriptor directly	
		09D8	09D8	1842			: Send database request to PSIACP
		09D8	09D8	1843			: Is there an active channel?
		00000000'EF	B5 09D8	1844	TSTW	NETSGW_X25_CHAN	: If NEQL then yes
		06 12	09DE	1845	BNEQ	60\$: Assign channel, get PSI mutex
		F61D'	30 09E0	1846	BSBW	NET\$GET_X25_CHAN	: If LBC then error
		3C 50	E9 09E3	1847	BLBC	R0,100\$: Re-issue QIO
		09E6	09E6	1848	\$QIOW_S	FUNC = #IOS_ACPCONTROL,-	: event flag for synchronous calls
		09E6	09E6	1849		EFN = #NETSC_EFN_WAIT,-	
		09E6	09E6	1850		CHAN = NETSGW_X25_CHAN,-	
		09E6	09E6	1851		IOSB = QIOW_Q_IOSB,-	: Scratch quadword buffer
				1852			
				1853			
				1854			

					P1 = 8(R3),-	; Address of NFB descriptor
					P2 = R2,-	; Address of P2 buffer descriptor
					P3 = #QIOW_W_RETLEN,-	; Address of return length word
					P4 = R3	; Address of P4 buffer
50	00000029'EF	OD 50	E9 0A12	1855 1859	BLBC R0,100\$	If LBC then error
		03 50	7D 0A15	1856 1860	MOVQ QIOW_Q_IOSB,R0	Setup IOSB image
		59 51	E8 0A1C	1857 1861	BLBS R0,100\$	Branch if successful
		5E 18	DO 0A1F	1858 1862	MOVL R1,R9	; Set error qualifier
			C0 0A22	1863 100\$:	ADDL #3+8,SP	; Deallocate descriptors
		00000029'EF	50 F5D1'	0A25 1864	MOVQ R0,QIOW_Q_IOSB	; Save final status
			30 0A2C	1865 1866	BSBW NET\$JNX_C0	Initialize journalling co-routine
			E9 0A2F	1866 1867	BLBC R0,105\$	Branch if not enabled
			90 0A32	1867 1868	MOVB #^X44,(R1)+	Journal record type = QIOW completion
81	00000029'EF	81 44 BF	B0 0A36	1868 1869	MOVW #IOS_ACPCONTROL,(R1)+	I/O function code
			7D 0A3B	1869 1870	MOVQ QIOW_Q_IOSB,(R1)+	I/O status quadword
			9E 16	1870 1871	JSB @(SP)+	Log the journalling record
50	00000029'EF	50 8ED0	7D 0A44	1871 1872	MOVQ QIOW_Q_IOSB,R0	; Restore final status
			0A4B	1872 105\$:		
			0A4B	1873 1874	BLBC R0,110\$; Branch if error detected
			22 66	1874 1875	CMPB NFB\$B_FCT(R6),#NFB\$C_FC_SHOW	; SHOW function?
			15 13	1875 1876	BEQL 150\$; Branch if so
			25 66	1876 1877	CMPB NFB\$B_FCT(R6),#NFB\$C_FC_ZERCOU	; READ & ZERO function?
			10 13	1877 1878	BEQL 150\$; Branch if so
			51 50	1878 1879	MOVL R0,R1	; Save final status
			8ED0 16	1879 110\$:	POPL R0	Retreive address of storage
			50 51	1880 1881	JSB NET\$DEALLOCATE	Deallocate scratch storage
			DO 0A58	1881 1882	MOVL R1,R0	Restore final status
			05 0A64	1882 1883	RSB	; Done
			0A67	1883 1884		
			0A68	1884 1885		
			0A68	1885 :		
			0A68	1886 :		
			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 descriptor of the returned buffer, and the		
			0A68	1889 : address of our scratch storage, which the caller will deallocate.		
			0A68	1890 :		
			0A68	1891 :		
57	00000031'EF	59 8ED0	0A68 1892	150\$:	POPL R9	; Return scratch pointer in R9
	58 012A C9	3C 0A6B	1893 1894		MOVZWL QIOW_W_RETLEN,R7	; Return descriptor of P4 buffer
		9E 0A72	1894 1895		MOVAB 12+NFB_SIZE+P2BUF_SIZE(R9),R8	; Exit successfully
		05 0A77			RSB	

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
 taken before marking a CNF for delete.
 0A78 1902 :
 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 :
 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,#NMASC_CIRTY_X25 ; X25 circuit?
 2C 12 0A8B 1921 BNEQ 10\$; Branch if not
 1C 50 E9 0A9A 1922 \$GETFLD cri_l_use ; Get X.25 usage
 00 58 D1 0A9D 1923 BLBC R0,10\$; If not specified, skip it
 17 12 0AA0 1924 CMPL R8,#NMASC_CIRUS_PER ; PVC?
 FEBF 30 0AA2 1925 BNEQ 10\$; Only ask PSI about PVCs
 13 50 E9 0AAC 1926 \$CNFFLD cri_s_nam,R9 ; Set field ID of circuit name
 66 21 90 0AAF 1927 BSBW SEND TO_PSI ; Call co-routine to setup NFB, P4, etc.
 9E 16 0AB2 1928 BLBC R0,90\$; Branch if error detected
 OB 50 E9 0AB4 1929 MOVB #NFBSC_FC_DELETE,NFB\$B_FCT(R6) ; Set function code
 06 11 0AB7 1930 JSB @(SP)+ ; Call SEND_TO_PSI to issue QIO
 0AB9 1931 BLBC R0,90\$; Branch if error detected
 0AB9 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

OAC6 1945 .SBTTL NET\$DELETE_PLI - Pre-delete processing
 OAC6 1946 :+
 OAC6 1947 : NET\$DELETE_PLI - Special processing before a PLI entry is marked for delete
 OAC6 1948 :
 OAC6 1949 : This routine is called to perform any special action that may need to be
 taken before marking a CNF for delete.
 OAC6 1950 :
 OAC6 1951 :
 OAC6 1952 : INPUTS: R11 CNR pointer
 OAC6 1953 : R10 CNF pointer
 OAC6 1954 :
 OAC6 1955 : OUTPUTS: R11,R10 Preserved
 OAC6 1956 : R0 LBS if successful
 OAC6 1957 : LBC if CNF should not be marked for delete
 OAC6 1958 :
 OAC6 1959 :
 OAC6 1960 :-
 OAC6 1961 :
 OAC6 1962 NET\$DELETE_PLI:
 08 50 E9 0AD3 1963 \$GETFLD pli,l,sta : Get the state value
 58 01 D1 0AD6 1964 BLBC R0,10\$: If LBC then no need to check state
 03 13 D4 0AD9 1965 CMPL #NMASC_STATE_OFF,R8 : Is the line turned off?
 50 05 D4 0ADB 1966 BEQL 10\$: If NEQ then cannot delete PLI
 05 05 0ADD 1967 CLRL R0 : Say "cannot delete PLI"
 0ADE 1968 RSB : Done
 0ADE 1969 :
 0ADE 1970 : If this is a LAPB line, ask PSIACP if it's ok to delete CNF
 0ADE 1971 :
 18 50 E9 0AEB 1972 10\$: \$GETFLD pli,l,pro : Get protocol parameter value
 05 58 D1 0AEE 1973 BLBC R0,DEAL_PLVEC : If not specified, skip it
 13 12 0AF1 1974 CMPL R8,#NMASC_LINPR_LAPB : LAPB line?
 FE3E 30 0AFA 1975 BNEQ DEAL_PLVEC : Branch if not
 05 50 E9 0AFD 1976 \$CNFFLD pli,s,nam,R9 : Set field ID of line name
 66 21 90 0B00 1977 BSBW SEND_TO_PSI : Call co-routine to setup NFB, P4, etc.
 9E 16 0B03 1978 BLBC R0,90\$: Branch if error detected
 05 05 0B05 1980 MOVBL #NFB\$C_FC_DELETE,NFB\$B_FCT(R6) : Set function code
 1981 90\$: JSB @(SP)+ : Call SEND_TO_PSI to issue QIO
 0B06 1982 RSB : Return with \$status.
 54 12 AA 3C 0B06 1983 DEAL_PLVEC: : Deallocate PLVEC cell
 1D 13 0B0A 1984 MOVZWL CNFSW_ID(R10),R4 : Get the PLVEC index
 00000000'EF44 94 0B0C 1985 BEQL 10\$: If EQL then done
 00000000'EF44 D4 0B13 1986 CLRBL PLVECSAB_REF[R4] : No longer any references to this slot
 0B1A 1987 CLRL PLVECSAL_UCB[R4] : No UCB
 0B1A 1988 \$DASSGN_S - : Deassign the I/O channel
 0B1A 1989 CHAN = PLVECSAW_CHAN[R4] :
 0B29 1990 :
 50 00' D0 0B29 1991 10\$: MOVL S^#SSS_NORMAL,R0 : Ignore errors
 05 0B2C 1992 RSB : Indicate success

0B2D 1994 .SBTTL NET\$REMOVE_xxx - Pre-remove 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

OB34 2012 .SBTTL CRI parameter action routines
 OB34 2013 :+
 OB34 2014 : NET\$CRI_V_LCK - Get status of conditionally writeable fields
 OB34 2015 :
 OB34 2016 : NET\$CRI_L_SUB - Get circuit's substate
 OB34 2017 : NET\$CRI_L_DRT - Get broadcast circuit's designated router
 OB34 2018 :
 OB34 2019 : NET\$CRI_L_PNA - Get partner's node address
 OB34 2020 : NET\$CRI_L_BLO - Get partner's receive block size
 OB34 2021 : NET\$CRI_L_LIT - Get partner's listen timer
 OB34 2022 :
 OB34 2023 : NET\$CRI_S_COL - Get circuit's collating value
 OB34 2024 : NET\$CRI_S_PNN - Get partner's node name
 OB34 2025 : NET\$CRI_S_LOO - Get line's loopback name
 OB34 2026 : NET\$CRI_S_CNT - Get (optionally clear) circuit counters
 OB34 2027 : NET\$CRI_S_VMSNAM Get VMS name of associated device
 OB34 2028 : NET\$CRI_S_CHR - Get circuit's characteristics buffer
 OB34 2029 :
 OB34 2030 : INPUTS: R11 CRI CNR address
 OB34 2031 : R10 CRI CNF address
 OB34 2032 : R9 FLD i.d. of field being read
 OB34 2033 : R1 Scratch
 OB34 2034 : R0 Scratch
 OB34 2035 :
 OB34 2036 : OUTPUTS: R1 Address of string descriptor or address of field value
 OB34 2037 : R0 Low bit set if R1 is valid
 OB34 2038 : Low bit clear otherwise
 OB34 2039 :
 OB34 2040 : All other register values are preserved.
 OB34 2041 :
 OB34 2042 :-
 OB34 2043 :
 F4C9' 30 2044 NET\$CRI_V_LCK:: ; Get status of cond. writeable fields
 07 50 E9 2045 B5BW NET\$LOCATE_LPD ; Locate the LPD
 1B A6 95 2046 BLBC R0,10\$; If LBC no LPD, not locked
 02 12 0B3A 2047 TSTB LPDSB_ASTCNT(R6) ; Any I/O outstanding?
 50 D4 0B3D 2048 BNEQ 10\$; If NEQ yes, locked
 51 50 D0 0B41 2049 CLRL R0 ; Else not locked
 50 00' D0 2050 10\$: MOVL R0,R1 ; Return as field value
 05 0B44 2051 MOVL S^#SS\$_NORMAL,RO ; Success
 0B47 2052 RSB ;
 0B48 2053 ;
 0B48 2054 ;
 F4B5' 30 2055 NET\$CRI_L_SUB:: ; Get circuit's substate
 0E 50 E9 2056 B5BW NET\$LOCATE_LPD ; Locate the LPD
 50 D4 0B4B 2057 BLBC R0,10\$; If LBC, no LPD or substate
 04 E0 0B50 2058 CLRL R0 ; Assume in "run" state
 51 07 22 A6 0B52 2059 BBS #LPDSV_RUN,- ; If BS then no sub-state
 27 A6 9A 0B55 2060 LPDSW_STS(R6),10\$;
 50 00' D0 0B59 2061 MOVZBL LPDSB_SUB_STA(R6),R1 ; Get sub-state
 05 0B5C 2062 MOVL S^#SS\$_NORMAL,RO ; Set status
 0B5D 2063 10\$: RSB ; Return to co-routine
 F4AO' 30 2065 NET\$CRI_L_DRT:: ; Get broadcast circuit's designated router
 25 50 E9 2066 B5BW NET\$LOCATE_LPD ; Locate the LPD
 50 D4 0B60 2067 BLBC R0,90\$; If not found, return "not set"
 0B63 2068 CLRL R0 ; Assume not a valid address

	1E 22 A6 OA	E1	OB65	2069	BBC	#LPDSV BC,LPDSW_STS(R6),90\$; No DRT if not broadcast circuit
	51 2C A6 18	3C	OB6A	2070	MOVZWL	LPDSW_DRT(R6),RT	; Get designated router ADJ index
54	00000000'EF	DO	OB70	2072	BEQL	90\$; If 0, then DRT "not set"
	51 2C B441	DO	OB77	2073	MOVL	NETSGL_PTR_VCB,R4	; Get RCB address
	51 04 A1 06	3C	OB7C	2074	MOVL	@RCBSL_PTR-ADJ(R4)[R1],R1	; Get ADJ address
	F47B' 50 00'	13 30	OB80 OB82	2075 2076	MOVZWL	ADJSW_PNA(R1),R1	; Get DRT node address
		DO	OB85	2077	BEQL	90\$; If 0, then DRT "not set"
		05	OB88	2078	BSBW	SUPPRESS_AREA	; Suppress area, if necessary
			OB89	2079	MOVL	S^#SSS_NORMAL,RO	; Set status
				90\$:	RSB		; Exit with status/value
	F474' 1E 50 50 04	30 E9 D4 E1	OB89 OB8C OB8F OB91	2080 2081 2082 2084	NET\$CRI_L_BLO:: BBC	NET\$LOCATE_LPD	; Get partner's receive block size
	17 22 A6 57 20 A6 00000000'EF	9A DO	OB93 OB96 OB9A	2085 2086 2087	BSBW BLBC CLRL BBC	RO,10\$ RO	; Locate the LPD
54	57 2C B447 51 06 A7 50 00'	DO	OB9A OBA1 OBA6 OBAAA	2087 2088 2089 2090	MOVZBL MOVL MOVL MOVL	LPDSB_PTH_INX(R6),R7 NET\$G[PTR_VCB,R4 @RCBSL_PTR-ADJ(R4)[R7],R7 ADJSW_BUFSIZ(R7),R1	; If LBC, no LPD or partner node
		05	OBAD	2091	10\$:	S^#SSS_NORMAL,RO RSB	; Assume not in "run" state
			OBAE	2092			; If BS then no block size
	F44F' 23 50 50 04	30 E9 D4 E1	OBAE OBB1 OBB4 OBB6	2093 2094 2095 2097	NET\$CRI_L_LIT:: BBC	NET\$LOCATE_LPD #LPDSV RUN,-	; Get LPD index
	1C 22 A6 0A 17 22 A6 57 20 A6 00000000'EF	9A DO	OB88 OBBB OBBD OBC0	2098 2099 2100 2101	BSBW BLBC CLRL BBC	LPDSW_STS(R6),10\$ RO,10\$ RO #LPDSV BC,-	; Get RCB address
54	57 2C B447 51 08 A7 50 00'	DO	OBBC OBC4 OBCB OBD0	2102 2103 2104	MOVZBL MOVL MOVL MOVL	LPDSB_PTH_INX(R6),R7 NET\$G[PTR_VCB,R4 @RCBSL_PTR-ADJ(R4)[R7],R7 ADJSW_INT[SN(R7),R1	; ; Get ADJ address
		05	OBD4 OBD7	2105 2106	10\$:	S^#SSS_NORMAL,RO RSB	; Get partner's listen timer
			OBD8	2107			; Set status
				2108			; Return to co-routine
	F425' 26 50 50 04	30 E9 D4 E1	OBD8 OBDB OBDE OBE0	2109 2110 2111 2112	NET\$CRI_L_PNA:: BBC	NET\$LOCATE_LPD #LPDSV BC,-	; Get partner's node address
	1F 22 A6 0A 1A 22 A6 57 20 A6 00000000'EF	9A DO	OBE2 OBE5 OBE7 OBEA	2113 2114 2115	BSBW BLBC CLRL BBC	LPDSW_STS(R6),10\$ RO,10\$ RO #LPDSV BC,-	; Locate the LPD
54	57 2C B447 51 04 A7 F3FF' 50 00'	DO	OBEE OBF5 OBF7 OBFA	2116 2117 2118	MOVZBL MOVL MOVL	LPDSB_PTH_INX(R6),R7 NET\$G[PTR_VCB,R4 @RCBSL_PTR-ADJ(R4)[R7],R7 ADJSW_PNA(R7),R1	; If LBC, no LPD or partner node
		05	OBFE	2120	BSBW	SUPPRESS_AREA	; Assume not in "run" state
		0C01	2121	2122	MOVL	S^#SSS_NORMAL,RO	; If BC then no partner node
		0C04	2123	10\$:	RSB		
		0C05	2124				
	F3F8' 30	OC05	2125	NET\$CRI_L_MST:: BBC	NET\$LOCATE_LPD		; Get maintenance mode state
							; Locate the LPD

51 00	E9 0C08	2126	BLBC R0,20\$: If LBC, no LPD or partner node
02	3C 0COB	2127	MOVZWL #NMASC_STATE_ON,R1	Assume state is "ON"
03 22 A6	E0 0COE	2128	BBS #LPDSV-DLE,-	If BS then maintenance state is "ON"
51 01	3C 0C10	2129	LPDSW_STS(R6),10\$	
50 00	D0 0C16	2130 10\$:	MOVZWL #NMASC STATE OFF,R1	: Else state is "OFF"
05	0C19	2131 20\$:	MOVL S^#\$\$\$_NORMAL,RO	: Set status
		2132	RSB	: Return to co-routine
		2133		
		2134		
		OC1A 2135	NET\$CRI_S_COL::	: Get circuit's collating value
		OC1A 2136	\$GETFLD cri_s_nam	: Get circuit's name
63 68 57	E9 0C27	2137	BLBC R0,10\$: Branch if not present
50 0000'8F	28 0C2A	2138	MOVC3 R7,(R8),(R3)	: Move the name
	3C 0C2E	2139	MOVZWL #\$\$\$_NORMAL,RO	: Indicate success
	05 0C33	2140 10\$:	RSB	: Return status in R0
		2141		
		OC34 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 #LPDSV_RUN,-	: If BC then no partner node
3F 22 A6	OC3E	2147	LPDSW_STS(R6),20\$	
0A	E0 0C41	2148	BBS #LPDSV_BC,-	: If a broadcast circuit,
3A 22 A6	OC43	2149	LPDSW_STS(R6),20\$: then no block size
57 20 A6	9A 0C46	2150	MOVZBL LPDSB_PTH_INX(R6),R7	: Get LPD index
54 00000000'EF	D0 0C4A	2151	MOVL NET\$GE_PTR_VCB,R4	: Get RCB address
57 2C B447	D0 0C51	2152	MOVL @RCBSL_PTR_ADJ(R4)[R7],R7	: Get ADJ address
58 04 A7	3C 0C56	2153	MOVZWL ADJSW_PNA(R7),R8	: Get partner's node address
5B 00000000'EF	D0 0C5A	2154	MOVL NET\$GE_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
09 50	E9 0C74	2157	\$GETFLD ndi_s_nna	: Get node name
63 68 57	28 0C77	2158	BLBC R0,20\$: If LBC then not set
50 0000'8F	3C 0C7B	2159	MOVC3 R7,(R8),(R3)	: Move the name
	05 0C80	2160 20\$:	MOVZWL #\$\$\$_NORMAL,RO	: Indicate success
		RSB		: Return status in R0
		OC81 2162		
		OC81 2163	NET\$CRI_S_L0O::	: Get line's loopback nodename
		OC81 2164	\$GETFLD cri_s_nam	: Get line name
5B 00000000'EF	E9 0C8E	2165	BLBC R0,20\$: If LBC then error
5A	D0 0C91	2166	MOVL NET\$GL_CNR_NDI,R11	: Get NDI root block
5A	D4 0C98	2167	CLRL R10	: Indicate no current NDI CNF
19 50	OC9A	2168	\$SEARCH egl_ndi,s,nli	: Get find NDI with this loopback line
19 50	E9 0CA9	2169	BLBC R0,20\$: If LBC then not found
09 50	0CAC	2170	\$GETFLD ndi_s_nna	: Get the node name
63 68 57	E9 0CB9	2171	BLBC R0,20\$: If LBC then not set
50 0000'8F	28 0CBC	2172	MOVC3 R7,(R8),(R3)	: Move the name
	3C 0CC0	2173	MOVZWL #\$\$\$_NORMAL,RO	: Indicate success
	05 0CC5	2174 20\$:	RSB	: Return status in R0
		0CC6 2175		
		0CC6 2176	NET\$CRI_S_VMSNAM::	: Get VMS name of device
03 50	E9 0CD3	2177	\$GETFLD cri_s_nam	: Get network management circuit name
0693	30 0CD6	2178	BLBC R0,10\$: If LBC then error
	0CD9	2179	BSBW TRAN_DEVNAM	: Get VMS device name
	05 0CD9	2180		: Exit with status
	05 0CD9	2181	: Assume that any remaining characters are part of tributary identifier	
		2182 100\$: RSB		: Return status in R0

		OCDA	2183		
		OCDA	2184	NET\$CRI_S_DEVNAM::	: Get VMS device name, including unit
	F323'	30	OCDA	2185 B5BW	: Locate LPD
	59 50	E9	OCDD	2186 BLBC	: Branch if none
	04	E1	OCEO	2187 BBC	: Exit if circuit not in run state
	54 22	A6	OCE2	2188 #LPDSV_RUN,-	
	5E 04	C2	OCE5	2189 LPDSW_STS(R6),80\$	
	54	5E	OCE8	2190 SUBL #4,SP	: Used for returning device characteristics
	00	DD	OCEB	2191 MOVL SP,R4	: Point R4 at return buffer
00000029'EF		9F	OCED	2192 PUSHL #0	: Terminate GETDVI item list
	54	DD	OCF3	2193 PUSHAB QIOW_Q_IOSB	: Point to DEVCHAR length word (use IOSB)
	7E 02	B0	OCF5	2194 MOVW #DVIS_DEVCHAR,-(SP)	: Point to DEVCHAR return buffer
	7E 04	B0	OCF8	2195 MOVW #4,-(SP)	: Item code
00000031'EF		9F	OCFB	2196 PUSHAB QIOW_W_RETLEN	: Size of DEVCHAR result buffer
	53	DD	OD01	2197 PUSHL R3	: Point to result length word
	7E 20	B0	OD03	2198 MOVW #DVIS_DEVNAM,-(SP)	: Point to result buffer
	7E 20	B0	OD06	2199 MOVW #DEVNAM_C_SIZE,-(SP)	: Item code
	51	5E	DO	2200 MOVL SP,R1	: Size of result buffer
			ODOC	2201 \$GETDVI_S_ITMLST=(R1),-	: Point to item list
			ODOC	2202 EFN=#NET\$C_EFN_WAIT,-	: Construct actual device name
			ODOC	2203 CHAN=LPDSW_CHAN(R6)	: Use synchronous event flag
	5E 20	C0	OD23	2204 ADDL #8*4,SP	
	15 50	E9	OD26	2205 BLBC R0,90\$: Pop item list
51	0C 64	0D	E1	2206 BBC #DEV\$V_NET,(R4),80\$: Branch if error detected
	53	51	OD29	2207 MOVZWL QIOW_W_RETLEN,R1	: Br if not a network device
	05	11	OD34	2208 ADDL R1,R3	: Get length of result buffer
50	0000'8F	3C	OD39	2209 BRB 90\$: Point past result string
			OD3E	2210 80\$: MOVZWL #SSS_IVDEVNAM,RO	: Success
			OD3E	2211 90\$: RSB	: Error in <cri,s,nam> field, or
			OD3F	2212 90\$: RSB	: not a network device
			OD3F	2213	: Return status in R0
	53	DD	OD3F	2214 NET\$CRI_S_CNT::	
			OD41	2215 PUSHL R3	: Get line counters (maybe clear them)
	31 50	E9	OD4E	2216 \$GETFLD cri_l_typ	: Save starting address of counter block
03	58	D1	OD51	2217 BLBC R0,20\$: Get circuit type
	2C	12	OD54	2218 CMPL R8,#NMASC_CIRTY_X25	: Branch if not set, assume not X25
	F2A7'	30	OD56	2219 BNEQ 20\$: X.25 circuit?
	21 50	E9	OD59	2220 BSBW NET\$LOCATE_LPD	: Branch if not
			OD5C	2221 BLBC R0,10\$: Get associated LPD, if any
			OD5C	2222	: Branch if no datalink mapping enabled
			OD5C	2223	
			OD5C	2224	
	3F	10	OD5C	2225 BSBW 50\$	
	53	DD	OD5E	2226 PUSHL R3	: Get Transport counters & seconds
	0065	30	OD60	2227 BSBW 200\$: Save current position in buffer
	58	8ED0	OD63	2228 POPL R8	: Get PSI circuit counters & seconds
OF 50		E9	OD66	2229 BLBC R0,5\$: Get start of PSI counter block
			OD69	2230	: Branch if no PSI counters (possible
			OD5C	2231	: for switched circuits)
C000 8F	68	B1	OD69	2232 CMPW (R8),#NET\$C_NMACNT_SLZ	: "seconds since last zeroed" ?
	1A	12	OD6E	2233 BNEQ 30\$: Branch if not
68	04	C2	OD70	2234 SUBL #4,R7	: Account for loss of PSI "seconds"
	57	28	OD73	2235 MOVC R7,4(R8),(R8)	: Delete PSI seconds counter from block
	50	01	DO	2235 5\$: MOVL #1,RO	: Success
	0D	11	OD7B	2236 BRB 30\$	
			OD7D	2237	
			OD7D	2238	
			OD7D	2239	
					: Native X.25 circuit counters

0048 08	30 11	0D7D 0D80	2240 2241	10\$: BSBW BRB	200\$ 30\$; Get PSI circuit counters & seconds ; and continue with status in R0	
		0D82	2242				
		0D82	2243		: Non-X25 circuit counters		
		0D82	2244				
F27B' 11 50	30 E9	0D82 0D85	2245 2246	20\$: BSBW BLBC	NET\$LOCATE_LPD R0,40\$; Get associated LPD ; Error if LBC	
13	10	0D88	2247		50\$; Get the Transport counters	
52 09 50	D0 E9	0D8A 0D8D	2248 2249	30\$: MOVL BLBC	(SP) R2 R0,40\$; Recover original counter block ptr ; If LBC then error	
50 03	D0	0D90	2250		S^#EVCS C SRC_CIR,R0	; Setup database event i.d.	
F26A' 50 00'	30 D0	0D93 0D96	2251 2252	MOVL BSBW	LOG COUNTERS S^#SSS_NORMAL,R0	; Log the counters if they were zeroed	
5E 04	C0	0D99	2253	40\$: MOVL ADDL	#4,SP #4,SP	; Success	
	05	0D9C	2254		RSB	; Pop buffer pointer	
		0D9D	2255			; Done	
		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+LPDS C_CNT_SIZE,R2	; Number of bytes of Transport counters	
		0DA7	2261			; +4 for ABS_TIME	
51 36 A6	9E	0DA7	2262	MOVAB	LPDSL_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				
		0DB1	2266		: Get the counters from the driver and append to current counters		
		0DB1	2267		; minus the 'seconds since last zeroed' counter.		
OE 22 A6 07	E0	0DB1	2268	BBS	#LPDSV_X25,LPDSW_STS(R6),90\$; Skip if X.25 datalink	
52 14 A6	3C	0DB6	2269	MOVZWL	LPDSW_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	#IOS_SENSEMODE!- IOSM_RD_COUNT,R0	; Setup function code	
50	ODC0	2272					
0673	30	0DC1	2273	BSBW	DEV_CNT_QIO	; Issue circuit QIO for counters	
50 00'	3C	0DC4	2274	90\$: MOVZWL	S^#SSS_NORMAL,R0	; Indicate success	
	05	0DC7	2275		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	
FB65	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	
03 00000000'EF	66 02	90 E1	0DE0 0DE3	MOVB	#NFBSC_FC SHOW,NFB\$B_FCT(R6)	; Set function code	
66 25	90	ODEB	2287 2288	BBC	#NET\$V-CLR_CNT,NET\$GL_FLAGS,55\$; Branch if not clearing counters	
53 00000200	8F	C0	0DEE	MOVB	#NFBSC_FC ZERO_COU,NFB\$B_FCT(R6)	; Set READ & ZERO function code	
	9E	16	0DF5	2289	55\$: ADDL	#P4BUF_SIZE,R3	; Set size of P4 return buffer
0048 8F	BA	0DF7	2290	JSB	a(SP)+	; Call SEND_TO_PSI back to issue QIO	
C9 50	E9	0DFB	2291	POPR	#^M<R3,R6>	; Restore registers	
57 88	3C	0DFE	2292	BLBC	R0,100\$; Branch if error detected	
63 68	28	0E01	2293	MOVZWL	(R8)+,R7	; Make descriptor of CNT string	
50 59	DO	0E05	2294	MOVC	R7,(R8),(R3)	; Copy P4 buffer into counter block	
00000000'EF	16	0E08	2295	MOVL	R9,R0	; Point to scratch storage	
			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::
 F1EB' 30 0E12 2301 B5BW NET\$LOCATE_LPD
 45 50 E9 0E15 2302 BLBC R0,100\$
 53 DD 0E18 2303 PUSHL R3
 00 6E 00 2C 0E1A 2304 MOVC5 #0,(SP),#0,-
 63 18 0E1E 2305 #DLLQIO\$C_LENGTH,(R3)
 55 8BED0 0E20 2306 POPL R5
 50 28 A6 9A 0E25 2307 CLRL R0
 32 13 0E29 2308 MOVZBL LPDSB_PLVEC(R6),R4
 50 10 A6 D0 0E2B 2310 BEQL 100\$
 2C 13 0E2F 2311 MOVL LPDSL_UCB(R6),R0
 0000'8F 3C 0E31 2312 BEQL 100\$
 65 0E35 2313 MOVZWL #IOS_SETMODE,-
 0E40 2314 DLLQIOSL_FUNC(R5)
 52 0000016C'EF 9E 0E36 2315 MOVAB CRI_TRN_TAB,R2
 5D'AF 9F 0E3D 2316 PUSHAB B^10\$
 0E40 2317 \$DISPATCH TYPE=B,PLVECSAB_DEV[R4],- ; Dispatch on device type
 0E40 2318 <-
 0E40 2319 <DEVTRNSC_DEV_DMC, CRI_DMC>,-
 0E40 2320 <DEVTRNSC_DEV_PCL, CRI_PCL>,-
 0E40 2321 <DEVTRNSC_DEV_DMP, CRI_DMP>,-
 0E40 2322 <DEVTRNSC_DEV_CI, CRI_CI>,-
 0E40 2323 <DEVTRNSC_DEV_UNA, CRI_UNA>,-
 01 11 0E5B 2324 BRB CRI_FOREIGN ; Br if device is unknown
 05 0E5D 2325 10\$: RSB
 0E5D 2326 100\$:
 0E5E 2327
 0E5E 2328 CRI_DMP:
 0E5E 2329 CRI_PCL:
 0E5E 2330 CRI_CI:
 0E5E 2331 CRI_UNA:
 044F 30 0E5E 2332 CRI_FOREIGN:
 05 0E61 2333 BSBW BUILD_DEVBUF ; Build the device parameter buffer
 0E62 2334 RSB ;
 0E62 2335
 04 A5 53 55 C3 0E62 2336 CRI_DMC:
 0E62 2337 SUBL3 R5,R3,DLLQIOSL_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 UCBSL_DEVDEPEND EQ 2+UCBSW_DEVBUFSIZ
 0E67 2346 ASSUME UCBSW_DEVBUFSIZ EQ 1+UCBSB_DEVTYPE
 0E67 2347 ASSUME UCBSB_DEVTYPE EQ 1+UCBSB_DEVCLASS
 0E67 2348
 83 40 A0 7D 0E67 2349 MOVQ UCBSB_DEVCLASS(R0), (R3)+; Enter device characteristics
 0E6B 2350 CLRBIT XMSV_CHR_MOP,-4(R3) ; Assume MOP mode not needed
 02 E1 0E6F 2351 BBC #LPDSV_DCE,- If BC not in use for direct-access
 04 22 A6 0E71 2352 LPDSW_STS(R6),10\$ functions
 0E74 2353 SETBIT XMSV_CHR_MOP,-4(R3) ; Put line in MOP mode

		OE78	2354	10\$:		
		OE78	2355			
		OE78	2356			Get number of receive buffers from the PLI database
		OE78	2357			
		OE78	2358			
5B	00000000'EF	BB	OE78	2359	PUSHR #^M<R1,R10,R11>	; Save regs
		DO	OE7C	2360	MOVL NET\$GL_CNR_PLI,R11	; Get PLI root block
	5A	D4	OE83	2361	CLRL R10	; Search from begining of list
58	54	DO	OE85	2362	MOVL R4,R8	; Search key is the PLVEC index
			OE88	2363	\$SEARCH eql,pli,l,plvec	Find PLI's CNF block
	10 50	E9	OE97	2364	BLBC R0,15\$; If LBC then not found
			OE9A	2365	\$GETFLD pli,l,bfn	; Get # of receive buffers
	03 50	E8	OEAA	2366	BLBS R0,20\$; If LBS then field was active
OC	58 04	DO	OEAD	2367 15\$:	MOVL #4,R8	; Setup default
	A5 58	DO	OEAD	2368 20\$:	MOVL R8,DLLQIO\$L P3(R5)	; Setup # of receive buffers
	OC02 8F	BA	OE81	2369	POPR #^M<R1,R10,R11>	; Restore regs
			OE85	2370		
	50 01	90	OE85	2371	MOVB #1,R0	; Indicate success
		05	OE88	2372 100\$:	RSB	; Return status in R0

0EB9 2374 .SBTTL PLI parameter action routines
 0EB9 2375 :+
 0EB9 2376 : NET\$PLI_V_LCK - Get status of conditionally writeable fields
 0EB9 2377 :
 0EB9 2378 : NET\$PLI_L_SUB - Get line's substate
 0EB9 2379 : NET\$PLI_L_BUS - Get line's receive buffer size
 0EB9 2380 : NET\$PLI_L_PLVEC Get line's PLVEC index
 0EB9 2381 :
 0EB9 2382 : NET\$PLI_S_COL - Get line's collating value
 0EB9 2383 : NET\$PLI_S_VMSNAM Get VMS name of associated device
 0EB9 2384 : NET\$PLI_S_DEVNAM Get VMS name of associated device, but with unit included
 0EB9 2385 : NET\$PLI_S_CNT - Get (optionally clear) line counters
 0EB9 2386 : NET\$PLI_S_CHR - Get line's characteristics buffer
 0EB9 2387 :
 0EB9 2388 : INPUTS: R11 PLI CNR address
 0EB9 2389 : R10 PLI CNF address
 0EB9 2390 : R9 FLD i.d. of field being read
 0EB9 2391 : R1 Scratch
 0EB9 2392 : R0 Scratch
 0EB9 2393 :
 0EB9 2394 : OUTPUTS: R1 Address of string descriptor or address of field value
 0EB9 2395 : R0 Low bit set if R1 is valid
 0EB9 2396 : Low bit clear otherwise
 0EB9 2397 :
 0EB9 2398 : All other register values are preserved.
 0EB9 2399 :
 0EB9 2400 :-
 0EB9 2401 :-
 0EB9 2402 NET\$PLI_V_LCK:: : Get status of cond. writeable fields
 0EB9 2403 \$GETFLD pli_l_sta : Get the state value
 58 07 50 E9 0EC6 2404 BLBC R0,10\$: If LBC then not locked
 01 D1 0EC9 2405 CMPL #NMASC_STATE_OFF,R8 : Is the line turned off?
 02 12 0ECC 2406 BNEQ 10\$: If NEQ then locked
 50 50 D4 0ECE 2407 CLRL R0 : Say "not locked"
 51 50 D0 0ED0 2408 10\$: MOVL R0,R1 : Return as field value
 50 00' D0 0ED3 2409 MOVL S^#SS\$_NORMAL,R0 : Success
 05 0ED6 2410 RSB :
 0ED7 2411 :
 50 D4 0ED7 2412 NET\$PLI_L_SUB:: : Get line's substate
 0ED9 2413 CRL R0 : No substate is defined for lines
 at this time
 05 0ED9 2414 RSB : Return status in R0
 0EDA 2415 :
 51 12 AA, 9A 0EDA 2417 NET\$PLI_L_PLVEC:: : Get line's PLVEC index
 50 00' D0 0EDE 2418 MOVZBL CNFSW_ID(R10),R1 : Get the PLVEC index
 05 0EE1 2419 MOVL S^#SS\$_NORMAL,R0 : Success
 0EE2 2420 RSB : Return
 0EE2 2421 :
 0EE2 2422 NET\$PLI_L_BUS:: : Get line's receive buffer size
 0EE2 2423 : (MUST preserve R2-R11)
 51 0380 8F BB 0EE2 2424 PUSHR #^M<R7,R8,R9> : Save registers
 00000000'EF D0 0EE6 2425 MOVL NET\$GL_PTR VCB,R1 : Get RCB address
 51 7E A1 3C 0EE6 2426 MOVZWL RCBSW_TOTB0FSIZ(R1),R1 : Store the local block size
 51 004C 8F A2 0EF1 2427 SUBW #CXB\$C_OVERHEAD,R1 : Subtract out overhead
 0EF6 2428 :
 0EF6 2429 : If this is an Ethernet controller, then add an additional
 0EF6 2430 : 15 bytes overhead to account for the extended Ethernet

5E 20 C0 0F95 2488 EFN=#NET\$C_EFN_WAIT,- ; Use synchronous event flag
 15 50 E9 0F95 2489 CHAN=R0
 0C 64 0D E1 0FAE 2490 ADDL #8*4 SP
 00000031'EF 3C 0FB1 2491 BLBC R0,90\$: Pop item list
 53 51 CO 0FB5 2492 BBC #DEVSV_NET,(R4),80\$: Branch if error detected
 05 11 OFBF 2493 MOVZWL QIOW_W_RETLEN,R1 : Br if not a network device
 50 0000'8F 3C 0FC1 2494 ADDL R1,R3 : Get length of result buffer
 0FC6 2495 BRB 90\$: Point past result string
 05 0FC6 2496 80\$: MOVZWL #SSS_IVDEVNAM,RO : Success
 0FC7 2497 90\$: RSB : Error in <pli,s,nam> field or
 0FC7 2499 : not a network device
 53 DD 0FC7 2500 NET\$PLI_S_CNT:: : Return status in R0
 0FC7 2501 PUSHL R3 : Get line counters (maybe clear them)
 0FC9 2502 : Save starting address of counter block
 0FC9 2503 :
 0FC9 2504 : If this is an X.25 circuit, then ask PSI for its counters.
 09 50 E9 0FD6 2505 \$GETFLD pli_l_pro : Get line protocol code
 58 58 D1 0FD9 2506 BLBC R0,10\$: Branch if not specified
 04 12 OFDC 2507 CMPL R8,#NMASC_LINPR_LAPB : LAPB line?
 55 10 OFDE 2508 BNEQ 10\$: If not, use normal SENSEMODE
 12 11 OFE0 2509 BSSB 200\$: Get PSI line counters & seconds
 0FE2 2510 BRB 20\$
 0FE2 2511 :
 0FE2 2512 : Get non-X25 line counters
 0FE2 2513 :
 56 12 AA 3C 0FE2 2514 10\$: MOVZWL CNFSW_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
 OFF0 2517 #DEVTRNSC_DEV_DMC : case this since XMDRIVER does not
 OE 13 OFF0 2518 BEQL 30\$: support the standard datalink driver
 OFF2 2519 : interface).
 13 10 OFF2 2520 BSSB 50\$: Get the counters
 52 6E D0 OFF4 2521 20\$: MOVL (SP),R2 : Recover original counter block ptr
 09 50 E9 OFF7 2522 BLBC R0,40\$: If LBC then error
 50 01 D0 OFFA 2523 MOVL S^#EVCS_C_SRC_LIN,RO : Setup database event i.d.
 F000' 30 OFFD 2524 BSBW LOG_COUNTERS : Log the counters if they were zeroed
 50 00' D0 1000 2525 30\$: MOVL S^#SS_NORMAL,RO : Success
 5E 04 CO 1003 2526 40\$: ADDL #4,SP : Pop counter address off stack
 05 1006 2527 RSB : Done
 1007 2528 :
 1007 2529 50\$: : Subroutine to build the counter block
 1007 2530 :
 1007 2531 :
 51 00000000'EF46 DE 1007 2532 MOVAL PLVEC\$AL_ABS_TIM[R6],R1 : Point to seconds since last zeroed
 52 04 D0 100F 2533 100F : counters
 55 00000028'EF 9E 1012 2534 MOVL #4,R2 : Number of bytes in block
 00000000'EF 16 1019 2535 MOVAB PLI_CNT_TAB,R5 : Point to counter formatting table
 101F 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.
 52 00000000'EF46 3C 101F 2541 70\$: MOVZWL PLVECSAW_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 #IOS_SENSEMODE!- : Setup function code
 102A 2544 IOSM_CTRL!-

10BF 2602 <DEVTRNSC_DEV_PCL, PLI_PCL>,-
 10BF 2603 <DEVTRNSC_DEV_DMP, PLI_DMP>,-
 10BF 2604 <DEVTRNSC_DEV_CI, PLI_CI>,-
 10BF 2605 <DEVTRNSC_DEV_UNA, PLI_UNA>,-
 10BF 2606 <DEVTRNSC_DEV_PPUNA, PLI_PPUNA>,-
 10BF 2607 >
 01 11 10DC 2608 BRB PLI_FOREIGN ; Br if device is unknown
 05 10DE 2609 40\$: RSB
 10DF 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 BO 10E8 2622 MOVW #NMASC_PCLI_PRM,(R3)+ ; Promiscuous mode = OFF
 83 01 DO 10ED 2623 MOVL #NMASC_STATE_OFF,(R3)+
 83 0B19 8F BO 10F0 2624 MOVW #NMASC_PCLI_MLT,(R3)+ ; Multicast address state = OFF
 83 01 DO 10F5 2625 MOVL #NMASC_STATE_OFF,(R3)+
 83 0B1A 8F BO 10F8 2626 MOVW #NMASC_PCLI_PAD,(R3)+ ; Padding on transmits = ON
 83 00 DO 10FD 2627 MOVL #NMASC_STATE_ON,(R3)+
 83 0B1B 8F BO 1100 2628 MOVW #NMASC_PCLI_DCH,(R3)+ ; Data chaining = OFF
 83 01 DO 1105 2629 MOVL #NMASC_STATE_OFF,(R3)+ ; (DLLTRN can't handle multiple XBs)
 83 0B1C 8F BO 1108 2630 MOVL #NMASC_PCLI_CRC,(R3)+ ; CRC generation = ON
 83 00 DO 110D 2631 MOVL #NMASC_STATE_ON,(R3)+
 83 0B1E 8F BO 1110 2632 MOVL #NMASC_PCLI_ACC,(R3)+ ; Protocol access mode = exclusive
 83 03 DO 1115 2633 MOVL #NMASC_ACC_EXC,(R3)+
 83 0B0E 8F BO 1118 2634 MOVL #NMASC_PCLI_PTY,(R3)+ ; Protocol type parameter code
 05 50 E8 112A 2635 \$GETFLD pli,l,ptp ; Get protocol type to be used
 58 0360 8F 3C 112D 2636 BLBS R0,10\$; Branch if ok
 83 58 DO 1132 2637 MOVZWL #TRSC_NI_PROT,R8 ; Else, provide a default value
 83 0B04 8F BO 1135 2638 10\$: MOVL R8,(R3)+ ; Set protocol type
 83 08 BO 113A 2640 MOVL #NMASC_PCLI_PHA,(R3)+ ; Physical address = Phase IV address
 83 01 BO 113D 2641 MOVL #8,(R3)+ ; 8 byte string:
 83 000400AA 8F DO 1140 2642 MOVL #NMASC_LINMC_SET,(R3)+ ; Set the address
 57 00000000'EF DO 1147 2643 MOVL #TRSC_NI_PREFIX,(R3)+ ; Phase IV NI prefix
 RCBSW PTR VCB,R7
 83 0E A7 BO 114E 2644 MOVL #NETSGE_ADDR(R7),(R3)+ ; Phase IV node number
 83 0BOF 8F BO 1152 2645 MOVL #NMASC_PCLI_MCA,(R3)+ ; Multicast addresses are:
 83 08 BO 1157 2646 MOVL #8,(R3)+ ; 8 byte string:
 83 01 BO 115A 2647 MOVL #NMASC_LINMC_SET,(R3)+ ; Set the address
 83 030000AB 8F DO 115D 2648 MOVL #TRSC_NI_ALLROUT1,(R3)+ ; Phase IV "all routers" multicast
 83 00 BO 1164 2649 MOVL #TRSC_NI_ALLROUT2,(R3)+
 05 008A C7 91 1167 2650 CMPB RCBSB_ETP(R7),#ADJSC_PTY_PH4N ; If Phase IV endnode,
 0D 12 116C 2651 BNEQ 20\$ use "all endnodes" instead
 83 040000AB 8F DO 1171 2652 SUBL #6,R3 of "all routers"
 83 06 C2 116E 2653 MOVL #TRSC_NI_ALLEND1,(R3)+ ; Phase IV "all endnodes" multicast
 83 00 BO 1178 2654 MOVL #TRSC_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,DLLQI0SL_P2(R5),R1 ; Point to P2 buffer descriptor
 61 52 C0 1184 2657 ADDL R2,(R1) ; Append these params to P2 buffer
 50 01 DO 1187 2658 MOVL #1,RO ; Success

05 118A 2659 RSB

118B 2660

118B 2661 PLI_PPUNA_TAB: : Parameter table for PPUNA pseudo-datalink

118B 2662 .cnvtab pcl1,pli,BUS : Receive buffer size

118F 2663 .cnvtab pcl1,pli,CON : Controller (loopback) mode

1193 2664 .cnvtab pcl1,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 BO 11A4 2671 MOVW #NMASC_PCLI_PRO,(R3)+ : Protocol type = POINT

83 00 DO 11A9 2672 MOVL #NMASC_LINPR_POI,(R3)+

83 0B18 8F BO 11AC 2673 MOVW #NMASC_PCLI_PRM,(R3)+

83 01 DO 11B1 2674 MOVL #NMASC_STATE_OFF,(R3)+

83 0B19 8F BO 11B4 2675 MOVW #NMASC_PCLI_MLT,(R3)+

83 01 DO 11B9 2676 MOVL #NMASC_STATE_OFF,(R3)+

83 0B1A 8F BO 11BC 2677 MOVW #NMASC_PCLI_PAD,(R3)+

83 00 DO 11C1 2678 MOVL #NMASC_STATE_ON,(R3)+

83 0B1B 8F BO 11C4 2679 MOVW #NMASC_PCLI_DCH,(R3)+

83 01 DO 11C9 2680 MOVL #NMASC_STATE_OFF,(R3)+

83 0B1C 8F BO 11CC 2681 MOVW #NMASC_PCLI_CRC,(R3)+

83 00 DO 11D1 2682 MOVL #NMASC_STATE_ON,(R3)+

83 0B1E 8F BO 11D4 2683 MOVW #NMASC_PCLI_ACC,(R3)+

83 02 DO 11D9 2684 MOVL #NMASC_ACC_IM,(R3)+

83 0B21 8F BO 11DC 2685 MOVW #NMASC_PCLI_DES,(R3)+

83 08 BO 11E1 2686 MOVW #8,(R3)+

83 01 BO 11E4 2687 MOVW #NMASC_LINMC_SET,(R3)+

83 00400AA 8F DO 11E7 2688 MOVL #TRSC_NI_PREFIX,(R3)+

00 58 0A ED 11EE 2689 \$GETFLD pli_l_ddf : Get value of dead timer

06 06 0A ED 11FB 2690 CMPZV #TR4SV_ADDR_AREA,- : Is area portion of address 0?

00 58 06 0E 12 1200 2691 BNEQ 5\$: If not, set it as is

57 00000000'EF 008B C7 DO 1202 2693 MOVL NET\$GL_PTR_VCB,R7 : Else, insert our own area

008B C7 FO 1209 2694 INSV RCB\$B_HOMEAREA(R7),-

0A 120D 2695 #TR4SV_ADDR_AREA,-

58 06 120E 2696 #TR4SS_ADDR_AREA,R8

83 58 BO 1210 2697 5\$: MOVW R8,(R3)+ : (leave R8 set for code below)

83 58 BO 1213 2698 MOVW #NMASC_PCLI_PTY,(R3)+ : Set remote destination address

83 0BOE 8F BO 1218 2699 \$GETFLD pli_l_ept : Protocol type parameter code

58 A002 8F 3C 1228 2700 BLBS R0,10\$: Get protocol type to be used

83 58 DO 122D 2701 MOVZWL #^XA002,R8 : Branch if ok

83 0B04 8F BO 1230 2702 10\$: MOVL R8,(R3)+ : Else, provide a default value

83 08 BO 1235 2704 MOVW #NMASC_PCLI_PHA,(R3)+ : Set protocol type

83 01 BO 1238 2705 MOVW #8,(R3)+ : Physical address = Phase IV address

83 00400AA 8F DO 123B 2706 MOVL #TRSC_NI_PREFIX,(R3)+ : 8 byte string:

57 00000000'EF DO 1242 2707 MOVL NET\$GL_PTR_VCB,R7 : Set the address

83 0E A7 BO 1249 2708 MOVW RCB\$B_ADDR(R7),(R3)+ : Phase IV NI prefix

52 53 8E C3 124D 2709 SUBL3 (SP)+,R3,R2 : Phase IV node number

51 08 A5 55 C1 1251 2710 ADDL3 R5,DLLQIOSL_P2(R5),R1 : Compute size of additional params

61 52 C0 1256 2711 ADDL R2,(R1) : Point to P2 buffer descriptor

50 01 DO 1259 2712 MOVL #1,R0 : Append these params to P2 buffer

05 125C 2713 RSB : Success

125D 2714

125D 2715 PLI_DMC:

			125D	2716	.IF NDF DMC_MODEM	
			125D	2717		; Disallow controller functions if
			125D	2718		; there is no modem support in DMC
		51 ED9E'	D4 30	125D 2719	CLRL R1	
		05 50	E8	125F 2720	BSBW NET\$GET_PLVECLPD	; Say "start with LPD index 0"
		0000'8F	AA	1262 2721	BLBS R0,10\$	Find next LPD actively using this line
		65		1265 2722	BICW #IOSM_SHUTDOWN,-	If LBS then found one
				1269 2723	DLLQI0\$L_FUNC(R5)	Can't issue "SHUTDOWN" if device is
			AA	126A 2724 10\$:	BICW #IOSM_STARTUP!-	not active
				126B 2725	IOSM_CTRL,-	Special case -- DMC does not support
65	0000'8F			126B 2726	DLLQI0\$L_FUNC(R5)	these modifiers in for its "line"
				126F 2727		; aspect
				126F 2728	.ENDC ; DF DMC_MODEM	
				126F 2729		
04 A5	53 55	C3	126F 2730	SUBL3 R5,R3,DLLQI0\$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			
	83 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 characteristics	
			1282 2739	\$GETFLD pli_v.con	; Get loopback field value	
	05 58	E9	128F 2740	BLBC R8,20\$; If LBC not in controller loopback	
	13	11	1292 2741	SETBIT XM\$V_CHR_LOOPB,(R3)	Else set loopback bit	
			1295 2742	BRB 30\$; Don't bother with HDPLX	
	03 58	E9	1297 2743 20\$:	\$GETFLD pli_v.dup	; Get duplex selector	
			12A4 2744	BLBC R8,30\$; LBS in R0 if half-duplex	
			12A7 2745	SETBIT XM\$V_CHR_HDPLX,(R3)	; Put line in half-duplex	
	50 83	D5	12AA 2746 30\$:	TSTL (R3)+	; Advance past characteristics	
	01	90	12AC 2747	MOVB #1,RO	; Indicate success	
	05	12AF	2748	RSB	; Return status in RO	

	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					
	08 A5 53 D0	2772	BUILD_DEVBUF:				
	12B0	2773	MOVL	R3,DLLQIOSL_P2(R5)	Build device param buffer		
	12B0	2774			Setup pointer to device parameter		
	12B0	2775	ADDL	#8,R3	buffer descriptor		
FC A3	08 B5 53 08 CE	12B0	2776	MNEGL	R3,@DLLQIOSL_P2(R5)	Allow 8 bytes for the descriptor	
	53 55 C3	12B0	2777	SUBL3	R5,R3,-4(R3)	Prepare for size calculation	
	0070 8F BB	12B0	2778	PUSHR	#^M<R4,R5,R6>	Setup offset to param buffer	
	56 52 D0	12B0	2779	MOVL	R2,R6	Save regs	
		12C7	2780			Point to translation table	
	66 D5	12C7	2781	TSTL	(R6)	At end of table?	
	45 13	12C9	2782	BEQL	40\$	If EQL then yes	
	3D 66 0C	12CB	2783	BBS	#CNVTABSV_R0,(R6),30\$	Skip if should not send to driver	
	7E OA AB	12CF	2784	MOV B	CNR\$B_TYPE(R11),-(SP)	Enter database i.d.	
	1E EF	12D3	2785	EXTZV	#CNVTABSV_FMT,-	Get param format type	
51	66 02	12D5	2786		#CNVTABSS_FMT,(R6),R1		
	7E 51 90	12D8	2787	MOV B	R1,-(SP)	Append to param database i.d.	
51	66 0E	12DB	2788	EXTZV	#CNVTABSV_INT,-	Get internal param i.d. index	
	7E 51 B0	12E0	2790	MOV W	#CNVTABSS_INT,(R6),R1		
	59 8ED0	12E3	2791	POPL	R1,-(SP)	Append param i.d.	
	59 B1	12E6	2792	CMPW	R9,#CNVTABSC_INTRNL	Get complete param i.d., fix stack	
	1F 13	12EB	2793	BEQL	30\$	Datalink only parameter?	
	ED10' 30	12ED	2794	BSBW	CNF\$GET_FIELD	If so, just ignore it	
	19 50 E9	12F0	2795	BLBC	R0,30\$	Get the field value	
	00 EF	12F3	2796	EXTZV	#CNVTABSV_NMA,-	If LBC then param is not set	
50	66 0C	12F5	2797		#CNVTABSS_NMA,(R6),R0	Get the NICE parameter code	
	83 50 B0	12F8	2798	MOV W	R0,(R3)+		
	02 51 D1	12FB	2799	CMPL	R1,#NFBSC_TYP_STR	Enter the NICE param i.d.	
	05 13	12FE	2800	BEQL	20\$	String ?	
	83 58 D0	1300	2801	MOVL	R8,(R3)+	If EQL yes	
	07 11	1303	2802	BRB	30\$	Enter the parameter value	
	83 57 B0	1305	2803	MOV W	R7,(R3)+	Continue	
63	68 57 28	1308	2804	MOVC3	R7,(R8),(R3)	Enter the string length	
	86 D5	130C	2805	30\$:	TSTL	Enter the string text, update R3	
	B7 11	130E	2806	BRB	(R6)+	Advance to next table entry	
					10\$	Loop	

```

      0070 8F    BA 1310 2807 40$:    POPR   #^M<R4,R5,R6>
      08 B5 53    CO 1314 2808          ADDL   R3,@DLLQI0$L_P2(R5)    ; Complete size calculation
      08 A5 55    C2 1318 2809          SUBL   R5,DLLQI0$L_P2(R5)    ; Convert pointer to offset
50  00000000'8F    D0 131C 2810          MOVL   #$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$CVT_NMA_INT:::
1324 2825 PUSRR #^M<R1,R2,R5> ; Convert NMA i.d. to internal i.d.
1324 2826 MOVAB CRI TRN TAB,R5 ; Save regs
1324 2827 CMPB #NFBSC DB CRI,- ; Assume CRI database
1324 2828 CNRSB_TYPE(R11) ; Circuit database ?
1324 2829 BEQL 10$ ; If EQL then yes
1324 2830 MOVAB PLI TRN TAB,R5 ; Else PLI database
1324 2831 10$: BSBB CVT_NMA_INT ; Convert the i.d.
1324 2832 POPR #^M<R1,R2,R5> ; Restore regs
1324 2833 RSB
1324 2834 :
1324 2835 :
1324 2836 : Do the actual conversion of NMA to NFB parameter codes.
1324 2837 :
1324 2838 : Inputs:
1324 2839 : R9 = NMA i.d. code, with format type
1324 2840 : R5 = Conversion table address
1324 2841 :
1324 2842 : Outputs:
1324 2843 : R1 = Last table entry used (zero if end of table)
1324 2844 : R0 = Status indicator
1324 2845 :
1324 2846 : R2 is destroyed.
1324 2847 :
1324 2848 :
1324 2849 CVT_NMA_INT:::
1324 2850 EXTZV #0,#CNVTABSS_NMA,R9,R2 ; Convert NMA i.d. to internal i.d.
1324 2851 CLRL R0 ; Get only NMA param i.d.
1324 2852 40$: MOVL (R5)+,R1 ; Assume conversion error
1324 2853 BEQL 100$ ; Get next table entry
1324 2854 :
1324 2855 ASSUME NFBSS_INX EQ 0 ; If EQL table is exhausted, report
1324 2856 ASSUME NFBSS_TYP EQ 16 ; that the driver is not supported
1324 2857 ASSUME NFBSS_DB EQ 24
1324 2858 :
1324 2859 CMPZV #CNVTAB$V_NMA,-
1324 2860 #CNVTAB$S_NMA,R1,R2 ; Is this it?
1324 2861 BNEQ 40$ ; If NEQ then keep trying
1324 2862 EXTZV #CNVTAB$V_INT,-
1324 2863 #CNVTAB$S_INT,R1,-(SP) ; Put internal code on stack longword
1324 2864 EXTZV #CNVTAB$V_FMT,- ; Get field format type
1324 2865 #CNVTAB$S_FMT,R1,R2
1324 2866 MOVB R2,(SP) ; Insert format into TYP field
1324 2867 MOVB CNRSB_TYPE(R11),3(SP) ; Insert database id into DB field
1324 2868 POPL R9 ; Get full field i.d., cleanup stack
1324 2869 MOVL #1,R0 ; Indicate successful conversion

```

NETCNFDLL
V04-000

F 4
- Datalink database action routines 16-SEP-1984 01:15:07 VAX/VMS Macro V04-00
NET\$CVT_NMA_INT - Convert NMA to NFB cod 5-SEP-1984 02:18:18 [NETACP.SRC]NETCNFDLL.MAR;1 Page 64
(28)
05 136B 2870 100\$: RSB

**F

136C 2872 .SBTTL TRAN_DEVNAM - Translate device name
 136C 2873 :+ TRAN_DEVNAM Translate Network Management name to VMS device name
 136C 2874 : This routine translates a network management line/circuit name to a VMS
 136C 2875 : device name. Network management line names have the format "dev-c-u" and
 136C 2876 : circuit names have the format "dev-c-u.t" where c,u,t are ascii-decimal
 136C 2877 : strings and "dev" is a network management device numeric.
 136C 2878 :
 136C 2879 :
 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 : OUTPUTS: 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 preserved
 136C 2894 :-
 136C 2895 TRAN_DEVNAM:
 136C 2896 :
 136C 2897 : Parse the device mnemonic, locate translation table entry
 136C 2898 :
 63 4B 4F 10 136C 2899 BSBB PRS_MNEMONIC ; Parse name, locate entry in table
 61 50 E9 136E 2900 BLBC R0,T10\$; If LBC then error
 08 0A A6 28 1371 2901 MOVC3 R2,(R1),(R3) ; Move the VMS device mnemonic
 33 13 1375 2902 CMPB DEVTRNS\$B_DEV(R6),#DEVTRNS\$C_DEV_X25 ; X25 circuit name?
 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)
 03 0C A6 00 E1 137E 2910 BBC #DEVTRNS\$V_MULTI,- ; If BC then not multi-unit device
 54 2D 9A 1380 2911 DEVTRNS\$B_CHAR(R6),20\$ and there's no unit field
 7E 10 1383 2912 MOVZBL #^A'-'',R4 ; Setup field delimiter
 31 50 E9 1388 2913 20\$: BSBB PRS_DECIMAL ; Convert the controller to binary
 52 D5 138B 2914 TSTL R2 ; If LBC then error
 28 13 138D 2915 BEQL 100\$ Any decimal characters parsed ?
 19 51 91 138F 2916 CMPB R1,#25 If EQL no, illegal name
 23 1A 1392 2917 BGTRU 100\$ Within range (0=A, 25=Z)
 83 51 41 8F 81 1394 2919 ADDB3 #^A'A'',R1,(R3)+ If GTRU then out of range
 1399 2920 : Store as VMS controller designator
 1399 2921 :
 1399 2922 : dev-c-u.t Parse the unit number.
 1399 2923 : We have parsed up to here
 50 D4 1399 2924 CLRL R0 ; Assume single-unit device
 00 E1 139B 2925 BBC #DEVTRNS\$V_MULTI,- ; If BC then single-unit device and
 0B 0C A6 54 2E 9A 13A0 2926 DEVTRNS\$B_CHAR(R6),30\$ there's no unit field
 61 10 13A3 2927 BSBB PRS_DECIMAL ; Setup field delimiter
 ; Convert the Ascii unit to binary

```
      14 50    E9 13A5 2929    BLBC   R0,110$ ; If LBC then illegal name
      51    D0 13AB 2930    MOVL   R1,R0 ; Get the binary value
  50 00000000'8F  EC52, 30 13AB 2931 30$: BSBW   NEFSBIN2ASC ; Move to @R3 as Ascii
      05    D0 13AE 2932 40$: MOVL   #SSS_NORMAL,RO ; Indicate success
          11 13B5 2933    BRB    110$ ; Take common exit
          13B7 2934
  50 0000'8F    3C 13B7 2935 100$: MOVZWL #SSS_IVDEVNAM,RO ; Error in translated name
      05 13BC 2936 110$:   RSB

```

```

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 SSS_NORMAL if successful
13BD 2953 : SSS_IVDEVNAM otherwise
13BD 2954 :
13BD 2955 :
13BD 2956 : All other regs are unchanged
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 #SSS_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 :
13D7 2972 ; Locate the table entry
56 0000001F'EF 9E 13D7 2973 MOVAB DEVTRN_TABLE-DEVTRNSC_LENGTH,R6 ; Prepare to scan device table
56 0D C0 13DE 2974 30$: ADDL #DEVTRNSC_LENGTH,R6 ; Advance to next entry
51 66 9A 13E1 2975 MOVZBL DEVTRNSB_NETMAN(R6),R1 ; Get length of network management name
51 16 13 13E4 2976 BEQL 40$ ; 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 DEVTRNST_NETMAN(R6) :
EC 12 13F0 2981 BNEQ 30$ ; If NEQ then loop
52 06 A6 9A 13F2 2982 MOVZBL DEVTRNSB_VMS(R6),R2 ; Setup length of VMS mnemonic
51 07 A6 9E 13F6 2983 MOVAB DEVTRNST_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^#SSS_NORMAL,R0 ; Indicate success
53 8ED0 1402 2987 100$: POPL R3 ; Restore R3
05 1405 2988 RSB ; Return status in R0

```

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 SSS_NORMAL if successful
 1406 3003 : SSS_INVDEVNAM otherwise
 1406 3004 :-
 1406 3005 PRS_DECIMAL:
 51 7C 1406 3006 CLRQ R1 ; Parse/convert ascii decimal to binary
 57 D7 1408 3007 10\$: DECL R7 ; Init binary value, character count
 1E 19 140A 3008 BLSS 15\$; Account for next character
 50 88 9A 140C 3009 MOVZBL (R8)+,R0 ; If LSS then done
 50 54 91 140F 3010 CMPB R4,R0 ; Get next character
 18 13 1412 3011 BEQL 20\$; Is it the delimiter?
 50 30 82 1414 3012 SUBB "#A'0',R0 ; If EQL then done
 18 19 1417 3013 BLSS 30\$; Convert to binary
 09 50 91 1419 3014 CMPB R0,#9 ; If LSS then not decimal
 13 1A 141C 3015 BGTRU 30\$; Decimal ?
 52 D6 141E 3016 INCL R2 ; If GTRU not decimal
 51 0A C4 1420 3017 MULL #10,R1 ; Bump number of digits
 0C 1D 1423 3018 BVS 30\$; Shift total one digit
 51 50 C0 1425 3019 ADDL R0,R1 ; If VS then R1 has overflowed
 DE 11 1428 3020 BRB 10\$; Complete the sum
 57 D4 142A 3021 15\$: CLRL R7 ; Loop
 50 00' 3C 142C 3022 20\$: MOVZWL S^#SSS_NORMAL,R0 ; Repair R7
 05 11 142F 3023 BRB 40\$; Indicate success
 50 0000'8F 3C 1431 3024 30\$: MOVZWL #SSS_INVDEVNAM,R0 ; Take common exit
 05 1436 3025 40\$: RSB ; Indicate invalid device name
 ; 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 SSS_NORMAL
 1437 3037 :
 1437 3038 : R1-R2,R4-R5 are destroyed.
 1437 3039 :-
 1437 3040 DEV_CNT_QIO:
 54 53 DO 1437 3041 MOVL R3,R4 : Save output buffer pointer
 02 E1 143A 3042 BBC #NETSV CLRcnt,- : If BC, don't clear the counters
 05 00000000'EF 143C 3043 NETSGL FLAGS,20\$
 50 0000'8F A8 1442 3044 BISW #IOSM 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),Z(R1),(R5) : Get ending buffer address
 65 53 C2 1460 3049 SUBL R3,(R5) : Setup bytes left in buffer
 04 A5 53 DO 1463 3050 MOVL R3,4(R5) : Point to it
 1467 3051 \$QIOW_S FUNC = R0,-
 1467 3052 CHAN = R2,-
 1467 3053 EFN = #NETSC_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),#NETSC_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 remainig counters, setup R3
 50 00 3C 14A6 3065 90\$: MOVZWL S^#SSS_NORMAL,R0 : Indicate success
 05 14A9 3066 RSB : Done
 14AA 3067
 14AA 3068 .END

- Datalink database action routines L 4

16-SEP-1984 01:15:07 VAX/VMS Macro V04-00
16-SEP-1984 02:18:18 [NETACP.SRC]NETCNFDLL.MAR;1Page 70
(32)

\$S\$DEVTRN	= 000000EF R 02	CRI_PSI_TAB	0000010C R 02
SST1	= 00000001	CRI_TO_PSI	00000573 R 04
SS_NSPMSG	= 00000000	CRI_TRN_TAB	0000016C R 02
SS_TR3MSG	= 00000000	CRI_UNA	00000E5E R 04
SS_TR4MSG	= 00000000	CVT_NMA_INT	0000133F R 04
SWIDTH_B	= 00000001	CXBSC_OVERHEAD	= 0000004C
SWIDTH_L	= 00000003	DCS_SCOM	= 00000020
SWIDTH_W	= 00000002	DEAL_PLVEC	00000B06 R 04
ACPSC_STA_F	= 00000004	DEFAULT_SCAN	***** X 04
ACPSC_STA_H	= 00000005	DEV\$V_NET	= 0000000D
ACPSC_STA_I	= 00000000	DEVNAM_C_SIZ	= 00000020
ACPSC_STA_N	= 00000001	DEVTRNSB_CHAR	= 0000000C
ACPSC_STA_R	= 00000002	DEVTRNSB_DEV	= 0000000A
ACPSC_STA_S	= 00000003	DEVTRNSB_NETMAN	= 00000000
ADJSC_PTY_AREA	= 00000003	DEVTRNSB_PROT	= 00000008
ADJSC_PTY_PH4	= 00000004	DEVTRNSB_VMS	= 00000006
ADJSC_PTY_PH4N	= 00000005	DEVTRNSC_DEV_CI	= 00000004
ADJSW_BUFSIZ	= 00000006	DEVTRNSC_DEV_DMC	= 00000001
ADJSW_INT_LSN	= 00000008	DEVTRNSC_DEV_DMF	= 00000003
ADJSW_PNA	= 00000004	DEVTRNSC_DEV_DMP	= 00000005
ALLOC_PLVEC	000006FF R 04	DEVTRNSC_DEV_DUP	= 00000006
BIT...	= 00000020	DEVTRNSC_DEV_KMS	= 00000007
BUILD_DEVBUF	00001280 R 04	DEVTRNSC_DEV_PCL	= 00000002
CCBSL_UCB	= 00000000	DEVTRNSC_DEV_PPUNA	= 0000000A
CLEAR_VOLATILE	00000360 R 04	DEVTRNSC_DEV_UNA	= 00000009
CNF\$B_FLG	= 0000000B	DEVTRNSC_DEV_UNK	= 00000000
CNF\$C[R_FIELD	***** X 04	DEVTRNSC_DEV_X25	= 00000008
CNF\$GET_FIELD	***** X 04	DEVTRNSC_LENGTH	= 0000000D
CNF\$KEY_SEARCH	***** X 04	DEVTRNSM_MULTI	= 00000001
CNF\$PUT_FIELD	***** X 04	DEVTRNST_NETMAN	= 00000001
CNF\$V_F[G_MRK1	= 00000004	DEVTRNST_VMS	= 00000007
CNF\$W_ID	= 00000012	DEVTRNSV_MULTI	= 00000000
CNFS_ADVANCE	= 00000000	DEVTRN_TABLE	0000002C R 02
CNFS_QUIT	= 00000002	DEV_CNT_QIO	00001437 R 04
CNFS_TAKE_CURR	= 00000003	DLLQIOSC_LENGTH	= 00000018
CNFS_TAKE_PREV	= 00000001	DLLQIOSL_FUNC	= 00000000
CNR\$B_TYPE	= 0000000A	DLLQIOSL_P1	= 00000004
CNVTAB\$C_INTRL	= 00003FFF	DLLQIOSL_P2	= 00000008
CNVTAB\$M_FMT	= C0000000	DLLQIOSL_P3	= 0000000C
CNVTAB\$M_INT	= 3FFF0000	DTS_DMC1T	= 00000001
CNVTAB\$M_NMA	= 00000FFF	DVIS_DEVCHAR	= 00000002
CNVTAB\$M_RO	= 00001000	DVIS_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	I0SM_CLR_COUNT	***** X 04
CNVTAB\$V_INT	= 00000010	I0SM_CTR	***** X 04
CNVTAB\$V_NMA	= 00000000	I0SM_RD_COUNT	***** X 04
CNVTAB\$V_RO	= 0000000C	I0SM_SHUTDOWN	***** X 04
CRI_CI	00000E5E R 04	I0SM_STARTUP	***** X 04
CRI_CLR_TAB	00000104 R 02	IOS_ACPCONTROL	***** X 04
CRI_DMC	00000E62 R 04	IOS_SENSEMODE	***** X 04
CRI_DMP	00000E5E R 04	IOS_SETMODE	***** X 04
CRI_FOREIGN	00000E5E R 04	IOCVERIFYCHAN	***** X 04
CRI_PCL	00000E5E R 04	JAM_CNF	000001E8 R 04
CRI_PSI_SIZ	= 00000020	LOG_COUNTERS	***** X 04

LPDSB_ASTCNT	=	0000001B		NETSC_MAX_OBJ	=	000000FF
LPDSB_CNT_IFL	=	0000004F		NETSC_MAX_WQE	=	00000014
LPDSB_CNT_LDN	=	0000004E		NETSC_MINBUFSIZ	=	000000C0
LPDSB_PLVEC	=	00000028		NETSC_NMACNT_SLZ	=	00000000
LPDSB_PTH_INX	=	00000020		NETSC_TID_ACT	=	00000003
LPDSB_SUB_STA	=	00000027		NETSC_TID_RUS	=	00000001
LPDSC_CNT_SIZE	=	00000016		NETSC_TID_XRT	=	00000002
LPDSL_ABS_TIM	=	00000036		NETSC_TRCTL_CEL	=	00000002
LPDSL_CNT_APX	=	0000003A		NETSC_TRCTL_OVR	=	00000005
LPDSL_CNT_DPS	=	00000042		NETSC_UTLBUFSIZ	=	00001000
LPDSL_CNT_TPR	=	0000003E		NET\$DEALLOCATE	*****	X 04
LPDSL_CNT_TPS	=	00000046		NET\$DECLARE_PSI	*****	X 04
LPDSL_UCB	=	00000010		NET\$DEFAULT_CRI	0000036B	RG 04
LPDSV_BC	=	0000000A		NET\$DEFAULT_PLI	00000420	RG 04
LPDSV_DLE	=	00000002		NET\$DELETE_CRI	00000A78	RG 04
LPDSV_RUN	=	00000004		NET\$DELETE_PLI	00000AC6	RG 04
LPDSV_X25	=	00000007		NET\$DLL_OPR_SET	*****	X 04
LPDSW_CHAN	=	00000014		NET\$GET_PLVECLFD	*****	X 04
LPDSW_CNT_ACL	=	0000004A		NET\$GET_X25_CHAN	*****	X 04
LPDSW_CNT_TCL	=	0000004C		NET\$GL_CNR_CNI	*****	X 04
LPDSW_DRT	=	0000002C		NET\$GL_CNR_NDI	*****	X 04
LPDSW_STS	=	00000022		NET\$GL_CNR_PLI	*****	X 04
LPD_CNT_TAB		00000000	R 02	NET\$GL_FLAGS	*****	X 04
MAX_C_DEVNAM				NET\$GL_PTR_LNI	*****	X 04
MOVE_FMT_CNT	*****		X 04	NET\$GL_PTR_VCB	*****	X 04
NET\$ALLOCATE	*****		X 04	NET\$GL_SRCH2_ID	*****	X 04
NET\$ALONPAGED	*****		X 04	NET\$GL_SRCH_ID	*****	X 04
NET\$APPLY_DFLT	*****		X 04	NET\$GQ_SRCH2_KEY	*****	X 04
NET\$BIN2ASC	*****		X 04	NET\$GQ_SRCHKEY	*****	X 04
NET\$CRI_L_BLO		00000B89	RG 04	NET\$GW_X25_CHAN	*****	X 04
NET\$CRI_L_DRT		00000B5D	RG 04	NET\$G_CRI_BDCMP	*****	X 04
NET\$CRI_L_LIT		00000BAE	RG 04	NET\$G_CRI_DLM	*****	X 04
NET\$CRI_L_MST		00000C05	RG 04	NET\$G_CRI_DLMOUT	*****	X 04
NET\$CRI_L_PNA		00000BD8	RG 04	NET\$G_CRI_NI	*****	X 04
NET\$CRI_L_SUB		00000B48	RG 04	NET\$G_CRI_TRN	*****	X 04
NET\$CRI_S_CHR		00000E12	RG 04	NET\$G_CRI_X25	*****	X 04
NET\$CRI_S_CNT		00000D3F	RG 04	NET\$G_PLI_DDCMP	*****	X 04
NET\$CRI_S_COL		00000C1A	RG 04	NET\$G_PLI_LAPB	*****	X 04
NET\$CRI_S_DEVNAM		00000CDA	RG 04	NET\$G_PLI_NI	*****	X 04
NET\$CRI_S_LOO		00000C81	RG 04	NET\$INSERT_CRI	000004BC	RG 04
NET\$CRI_S_PNN		00000C34	RG 04	NET\$INSERT_PLI	000005EF	RG 04
NET\$CRI_S_VMSNAM		00000CC6	RG 04	NET\$JNX_CO	*****	X 04
NET\$CRI_V_LCK		00000B34	RG 04	NET\$LOCATE_LPD	*****	X 04
NET\$CVT_NMA_INT		00001324	RG 04	NET\$M_MAXLNKMSK	= 000003FF	X 04
NET\$ACT_TIMER	=	0000001E		NET\$NDI_BY_ADD	*****	X 04
NET\$EFN_ASYN	=	00000002		NET\$PLI_L_BUS	00000EE2	RG 04
NET\$EFN_WAIT	=	00000001		NET\$PLI_L_PLVEC	00000EDA	RG 04
NET\$C_IPL	=	00000008		NET\$PLI_L_SUB	00000ED7	RG 04
NET\$C_MAXACFLD	=	00000027		NET\$PLI_S_CHR	0000107F	RG 04
NET\$C_MAXLINNAM	=	0000000F		NET\$PLI_S_CNT	00000FC7	RG 04
NET\$C_MAXLNK	=	000003FF		NET\$PLI_S_COL	00000F26	RG 04
NET\$C_MAXNODNAM	=	00000006		NET\$PLI_S_DEVNAM	00000F60	RG 04
NET\$C_MAXOBJNAM	=	0000000C		NET\$PLI_S_VMSNAM	00000F40	RG 04
NET\$C_MAX_AREAS	=	0000003F		NET\$PLI_V_LCK	00000EB9	RG 04
NET\$C_MAX_LINES	=	00000040		NET\$PRE_QIO_CRI	00000003	RG 04
NET\$C_MAX_NCB	=	0000006E		NET\$PRE_QIO_PLI	0000000C	RG 04
NET\$C_MAX_NODES	=	000003FF		NET\$REMOVE_CRI	00000B2D	RG 04

NET\$REMOVE_DEF
 NET\$REMOVE_PLI
 NET\$SCAN_CRI
 NET\$SCAN_PLI
 NET\$SET_CTR_TIMER
 NET\$SET_QIOW
 NET\$SHOW_CRI
 NET\$SHOW_PLI
 NET\$TABLE_DFLT
 NET\$V_CLR_CNT
 NET\$V_CNFLCK
 NFB\$B_DATABASE
 NFB\$B_FCT
 NFB\$B_FLAGS
 NFB\$B_OPER
 NFBSC_CRI_ACB
 NFBSC_CRI_ACI
 NFBSC_CRI_BBT
 NFBSC_CRI_CHN
 NFBSC_CRI_CNT
 NFBSC_CRI_DLM
 NFBSC_CRI_DTE
 NFBSC_CRI_DTH
 NFBSC_CRI_DYB
 NFBSC_CRI_DYI
 NFBSC_CRI_DYT
 NFBSC_CRI_IAB
 NFBSC_CRI_IAI
 NFBSC_CRI_IAT
 NFBSC_CRI_MBL
 NFBSC_CRI_MRB
 NFBSC_CRI_MST
 NFBSC_CRI_MTR
 NFBSC_CRI_MWI
 NFBSC_CRI_NAM
 NFBSC_CRI_PLS
 NFBSC_CRI_POL
 NFBSC_CRI_SER
 NFBSC_CRI_STA
 NFBSC_CRI_TRI
 NFBSC_CRI_TRT
 NFBSC_CRI_TYP
 NFBSC_CRI_USE
 NFBSC_CRI_VMSNAM
 NFBSC_CTX_SIZE
 NFBSC_DB_CRI
 NFBSC_ENDOFLIST
 NFBSC_FC_DELETE
 NFBSC_FC_SET
 NFBSC_FC_SHOW
 NFBSC_FC_ZERCOU
 NFBSC_LENGTH
 NFBSC_NDI_NLI
 NFBSC_NDI_NNA
 NFBSC_OP_EQL
 NFBSC_PLI_BFN
 NFBSC_PLI_BFS

***** X 04	NFBSC_PLI_BUS	= 0501001F
00000B2D RG 04	NFBSC_PLI_CHR	= 05020043
00000000 RG 04	NFBSC_PLI_CLO	= 05000005
00000000 RG 04	NFBSC_PLI_CNT	= 05020044
***** X 04	NFBSC_PLI_CON	= 05000004
00000844 RG 04	NFBSC_PLI_DDT	= 0501001B
00000118 RG 04	NFBSC_PLI_DLT	= 0501001C
0000015C RG 04	NFBSC_PLI_DUP	= 05000003
***** X 04	NFBSC_PLI_EPT	= 05010026
= 00000002	NFBSC_PLI_HTI	= 05010016
= 0000000B	NFBSC_PLI_HWA	= 05020046
= 00000002	NFBSC_PLI_MBL	= 05010017
= 00000000	NFBSC_PLI_MCD	= 05020045
= 00000001	NFBSC_PLI_MOD	= 05010022
= 00000003	NFBSC_PLI_MRT	= 05010018
= 04010029	NFBSC_PLI_MWI	= 05010019
= 0401002A	NFBSC_PLI_NAM	= 05020041
= 04010025	NFBSC_PLI_PLVEC	= 05010020
= 04010021	NFBSC_PLI_PRO	= 05010014
= 04020044	NFBSC_PLI_RTT	= 05010021
= 04000005	NFBSC_PLI_SLT	= 0501001A
= 04020049	NFBSC_PLI_SRT	= 0501001D
= 04010031	NFBSC_PLI_STA	= 05010011
= 0401002E	NFBSC_PLI_SUB	= 05010012
= 0401002F	NFBSC_TYP_STR	= 00000002
= 04010030	NFB\$L_FLDID	= 00000010
= 0401002B	NFB\$L_SRCH_KEY	= 00000004
= 0401002C	NFB\$M_INX	= 0000FFFF
= 0401002D	NFB\$M_NOCTX	= 00000004
= 04010022	NFB\$M_TYP	= 00030000
= 04010027	NFB\$S_TYP	= 00000002
= 04010032	NFB\$V_DB	= 00000018
= 04010028	NFB\$V_INX	= 00000000
= 04010023	NFB\$V_TYP	= 00000010
= 04020041	NFB_SIZE	= 00000088
= 0401001E	NMASC_ACC_EXC	= 00000003
= 0401001D	NMASC_ACC_LIM	= 00000002
= 04000002	NMASC_CIRTY_X25	= 00000003
= 04010013	NMASC_CIRUS_INC	= 00000001
= 04010024	NMASC_CIRUS_OUT	= 00000002
= 04010026	NMASC_CIRUS_PER	= 00000000
= 04010020	NMASC_CTCIR_ACL	= 00000322
= 0401001F	NMASC_CTCIR_APP	= 00000320
= 04020042	NMASC_CTCIR_DPS	= 00000321
= 00000040	NMASC_CTCIR_IFL	= 00000335
= 00000004	NMASC_CTCIR_LDN	= 00000334
= 00000000	NMASC_CTCIR_TCL	= 0000032C
= 00000021	NMASC_CTCIR_TPR	= 0000032A
= 00000023	NMASC_CTCIR_TPS	= 0000032B
= 00000022	NMASC_LINMC_SET	= 00000001
= 00000025	NMASC_LINPR	= FFFFFFFF
= 00000010	NMASC_LINPR_CON	= 00000001
= 0202004C	NMASC_LINPR_DMC	= 00000004
= 02020043	NMASC_LINPR_LAPB	= 00000005
= 00000000	NMASC_LINPR_MAS	= 00000001
= 0501001E	NMASC_LINPR_NI	= 00000006
= 05010027	NMASC_LINPR_POI	= 00000000

NMASC_LINPR_TRI	= 00000002	NSPSC_HSZ_ACK	= 00000007
NMASC_PCCI_ACB	= 0000047E	NSPSC_HSZ_CA	= 00000003
NMASC_PCCI_ACI	= 0000047F	NSPSC_HSZ_CC	= 00000064
NMASC_PCCI_BBT	= 00000475	NSPSC_HSZ_CD	= 000000F0
NMASC_PCCI_DTH	= 00000486	NSPSC_HSZ_CI	= 000000F0
NMASC_PCCI_DYB	= 00000483	NSPSC_HSZ_DATA	= 00000009
NMASC_PCCI_DYI	= 00000484	NSPSC_HSZ_DC	= 00000016
NMASC_PCCI_DYT	= 00000485	NSPSC_HSZ_DI	= 00000016
NMASC_PCCI_IAB	= 00000480	NSPSC_HSZ_INT	= 00000009
NMASC_PCCI_IAI	= 00000481	NSPSC_HSZ_LS	= 00000009
NMASC_PCCI_IAT	= 00000482	NSPSC_INF_V31	= 00000001
NMASC_PCCI_MRB	= 00000479	NSPSC_INF_V32	= 00000000
NMASC_PCCI_MST	= 00000AFA	NSPSC_INF_V33	= 00000002
NMASC_PCCI_MTR	= 0000047A	NSPSC_MAXHDR	= 00000009
NMASC_PCCI_PLS	= 000003F3	NSPSC_MSG_CA	= 00000024
NMASC_PCCI_POL	= 000003F2	NSPSC_MSG_CC	= 00000028
NMASC_PCCI_TRI	= 00000474	NSPSC_MSG_CI	= 00000018
NMASC_PCCI_TRT	= 00000476	NSPSC_MSG_DATA	= 00000000
NMASC_PCCI_TYP	= 00000458	NSPSC_MSG_DC	= 00000048
NMASC_PCLI_ACC	= 00000B1E	NSPSC_MSG_DI	= 00000038
NMASC_PCLI_BFN	= 00000451	NSPSC_MSG_DTACK	= 00000004
NMASC_PCLI_BUS	= 00000AF1	NSPSC_MSG_INT	= 00000030
NMASC_PCLI_CLO	= 00000459	NSPSC_MSG_LIACK	= 00000014
NMASC_PCLI_CON	= 00000456	NSPSC_MSG_LS	= 00000010
NMASC_PCLI_CRC	= 00000B1C	NSPSC_SRV_MFC	= 00000002
NMASC_PCLI_DCH	= 00000B1B	NSPSC_SRV_NFC	= 00000000
NMASC_PCLI_DDT	= 0000047F	NSPSC_SRV_REQ	= 00000001
NMASC_PCLI_DES	= 00000B21	NSPSC_SRV_SFC	= 00000001
NMASC_PCLI_DLX	= 00000480	NSPSM_ACK_NAK	= 00001000
NMASC_PCLI_DUP	= 00000457	NSPSM_ACK_NUM	= 0000FFFF
NMASC_PCLI_HWA	= 00000488	NSPSM_ACK_VALID	= 00008000
NMASC_PCLI_MCA	= 00000B0F	NSPSM_DATA_BOM	= 00000020
NMASC_PCLI_MLT	= 00000B19	NSPSM_DATA_EOM	= 00000040
NMASC_PCLI_PAD	= 00000B1A	NSPSM_DATA_OVFW	= 00000080
NMASC_PCLI_PHA	= 00000B04	NSPSM_FLW_CHAN	= 0000000C
NMASC_PCLI_PRM	= 00000B18	NSPSM_FLW_DRV	= 000000F0
NMASC_PCLI_PRO	= 00000458	NSPSM_FLW_INT	= 00000020
NMASC_PCLI_PTY	= 00000B0E	NSPSM_FLW_INUSE	= 00000010
NMASC_PCLI_RTT	= 00000461	NSPSM_FLW_LISUB	= 00000004
NMASC_PCLI_SLT	= 0000047E	NSPSM_FLW_MODE	= 00000003
NMASC_PCLI_SRT	= 00000481	NSPSM_FLW_SP1	= 00000008
NMASC_STATE_OFF	= 00000001	NSPSM_FLW_SP2	= 00000040
NMASC_STATE_ON	= 00000000	NSPSM_FLW_SP3	= 00000080
NMASC_STATE_SER	= 00000002	NSPSM_FLW_XOFF	= 00000001
NSPSSS_QUAL_ACK	= 00000000	NSPSM_FLW_XON	= 00000002
NSPSSS_QUAL_ALTFW	= 00000000	NSPSM_INF_VER	= 00000003
NSPSSS_QUAL_DATA	= 00000000	NSPSM_MSG_INT	= 00000020
NSPSSS_QUAL_FLW	= 00000000	NSPSM_MSG_LI	= 00000010
NSPSSS_QUAL_INF	= 00000000	NSPSM_SRV_01	= 00000003
NSPSSS_QUAL_MSG	= 00000000	NSPSM_SRV_EXT	= 00000080
NSPSSS_QUAL_SRV	= 00000000	NSPSM_SRV_FLW	= 0000000C
NSPSC_EXT_LNK	= 0000001E	NSPSM_SRV_REQ	= 000000F3
NSPSC_FLW_DATA	= 00000000	NSPSM_SRV_SP1	= 00000070
NSPSC_FLW_INT	= 00000001	NSPSR_QUA[= 00000000
NSPSC_FLW_NOP	= 00000000	NSPSS_ACK_NUM	= 0000000C
NSPSC_FLW_XOFF	= 00000001	NSPSS_ACK_SP2	= 00000002
NSPSC_FLW_XON	= 00000002	NSPSS_DATA_SP	= 00000005

NSP\$S_FLW_CHAN	= 00000002		PLI_PPUNA_TAB	= 0000118B R 04
NSP\$S_FLW_DRV	= 00000004		PLI_PSI_SIZ	= 00000038
NSP\$S_FLW_MODE	= 00000002		PLI_PSI_TAB	= 00000134 R 02
NSP\$S_INF_VER	= 00000002		PLI_Q_DEVNAM	= 00000001 R 03
NSP\$S_MSG_SP1	= 00000004		PLI_TO_PSI	= 000008F1 R 04
NSP\$S_NSPMSG	= 00000005		PLI_TRN_TAB	= 000001B8 R 02
NSP\$S_QUAL	= 00000005		PLI_UNA	= 000010E3 R 04
NSP\$S_QUAL_ACK	= 00000002		PLI_V_NEW	= 00000004
NSP\$S_QUAL_ALTFWL	= 00000001		PLVECSAB_DEV	***** X 04
NSP\$S_QUAL_DATA	= 00000001		PLVECSAB_REF_C	***** X 04
NSP\$S_QUAL_FLW	= 00000001		PLVECSAB_STATE	***** X 04
NSP\$S_QUAL_INF	= 00000001		PLVECSAL_ABS_TIM	***** X 04
NSP\$S_QUAL_MSG	= 00000005		PLVECSAL_UCB	***** X 04
NSP\$S_QUAL_SRV	= 00000001		PLVECSAW_CHAN	***** X 04
NSP\$S_SRV_01	= 00000002		PLVECSGB_MAX	***** X 04
NSP\$S_SRV_FLW	= 00000002		PREQIO_PRI_CRI	00000013 R 04
NSP\$S_SRV_SP1	= 00000003		PRS_DECIMAL	00001406 R 04
NSP\$V_ACK_NAK	= 0000000C		PRS_MNEMONIC	000013BD R 04
NSP\$V_ACK_NUM	= 00000000		PSI_PLI_CLR_TAB	0000012C R 02
NSP\$V_ACK_SP2	= 0000000D		QIOW_Q_IOSB	00000029 R 03
NSP\$V_ACK_VALID	= 0000000F		QIOW_W_RETLEN	00000031 R 03
NSP\$V_DATA_BOM	= 00000005		RCBSBETY	= 0000008A
NSP\$V_DATA_EOM	= 00000006		RCBSB_HOMEAREA	= 0000008B
NSP\$V_DATA_OVFW	= 00000007		RCBSL_PTR_ADJ	= 0000002C
NSP\$V_DATA_SP	= 00000000		RCBSW_ADDR	= 0000000E
NSP\$V_FLW_CHAN	= 00000002		RCBSW_TOTBUFSIZ	= 0000007E
NSP\$V_FLW_DRV	= 00000004		REBUILDED_NAME	0000007B R 04
NSP\$V_FLW_INT	= 00000005		SEND_TO_PSI	0000093B R 04
NSP\$V_FLW_INUSE	= 00000004		SHO_PLI_CRI	0000019E R 04
NSP\$V_FLW_LISUB	= 00000002		SIZ...	= 00000002
NSP\$V_FLW_MODE	= 00000000		SRCH_BUF	00000009 R 03
NSP\$V_FLW_SP1	= 00000003		SSS_BADPARAM	***** X 04
NSP\$V_FLW_SP2	= 00000006		SSS_INSFARG	***** X 04
NSP\$V_FLW_SP3	= 00000007		SSS_INSFMEM	***** X 04
NSP\$V_FLW_XOFF	= 00000000		SSS_IVDEVNAM	***** X 04
NSP\$V_FLW_XON	= 00000001		SSS_NORMAL	***** X 04
NSP\$V_INF_VER	= 00000000		SSS_NOSUCHDEV	***** X 04
NSP\$V_MSG_INT	= 00000005		SSS_RESULTOVF	***** X 04
NSP\$V_MSG_LI	= 00000004		SSS_WRITLCK	***** X 04
NSP\$V_MSG_SP1	= 00000000		STORE_PSI_PARAMS	000002F4 R 04
NSP\$V_SRV_01	= 00000000		SUPPRESS_AREA	***** X 04
NSP\$V_SRV_EXT	= 00000007		SYSSASSIGN	***** GX 04
NSP\$V_SRV_FLW	= 00000002		SYSSDASSGN	***** GX 04
NSP\$V_SRV_SP1	= 00000004		SYSSGETDVI	***** GX 04
NSP\$W_DSTLNK	= 00000001		SYSSQIOW	***** GX 04
NSP\$W_SRCLNK	= 00000003		TAKE_2_IOSB	00000270 R 04
P2BUF_SIZE	= 00000066		TAKE_R7_IOSB	00000273 R 04
P4BUF_SIZE	= 00000200		TMPBUF_DESC	***** X 04
PLI_BSTATE	000000000 R 03		TMP_Q_DESC	00000033 R 03
PLI_CI	000010DF R 04		TRSC_MAXHDR	= 0000001C
PLI_CLR_TAB	000000FC R 02		TRSC_NI_ALLEND1	= 040000AB
PLI_CNT_TAB	00000028 R 02		TRSC_NI_ALLEND2	= 00000000
PLI_DMC	0000125D R 04		TRSC_NI_ALLROUT1	= 030000AB
PLI_DMP	000010DF R 04		TRSC_NI_ALLROUT2	= 00000000
PLI_FOREIGN	000010DF R 04		TRSC_NI_PREFIX	= 000400AA
PLI_PCL	000010DF R 04		TRSC_NI_PROT	= 00000360
PLI_PPUNA	0000119B R 04		TRSC_PRTECL	= 0000001F

TR\$C_PRI_RTHRU	= 0000001F	TR4\$R_QUAL	= 00000000
TR3\$\$S_QUAL_MSG	= 00000000	TR4\$S_ADDR_AREA	= 00000006
TR3\$\$S_QUAL_RTFLG	= 00000000	TR4\$S_ADDR_DEST	= 0000000A
TR3\$C_RSZ_DATA	= 00000006	TR4\$S_QUAL	= 00000002
TR3\$C_MSG_DATA	= 00000002	TR4\$S_QUAL_ADDR	= 00000002
TR3\$C_MSG_HELLO	= 00000005	TR4\$S_QUAL_RTFLG	= 00000001
TR3\$C_MSG_INIT	= 00000001	TR4\$S_QUAL_SCLASS	= 00000001
TR3\$C_MSG_NOP2	= 00000008	TR4\$S_RTFLG_01	= 00000002
TR3\$C_MSG_ROUT	= 00000007	TR4\$S_RTFLG_VER	= 00000002
TR3\$C_MSG_STR2	= 00000058	TR4\$S_SCLASS_57	= 00000003
TR3\$C_MSG_VERF	= 00000003	TR4\$S_TR4MSG	= 00000002
TR3\$M_MSG_CTL	= 00000001	TR4\$V_ADDR_AREA	= 0000000A
TR3\$M_MSG_RTH	= 00000002	TR4\$V_ADDR_DEST	= 00000000
TR3\$M_RTFLG_PH2	= 00000040	TR4\$V_RTFLG_01	= 00000000
TR3\$M_RTFLG_RQR	= 00000008	TR4\$V_RTFLG_INI	= 00000005
TR3\$M_RTFLG_RTS	= 00000010	TR4\$V_RTFLG_LNG	= 00000002
TR3\$R_QUAL	= 00000000	TR4\$V_RTFLG_RQR	= 00000003
TR3\$S_QUAL	= 00000001	TR4\$V_RTFLG_RTS	= 00000004
TR3\$\$S_QUAL_MSG	= 00000001	TR4\$V_RTFLG_VER	= 00000006
TR3\$\$S_QUAL_RTFLG	= 00000001	TR4\$V_SCLASS_1	= 00000001
TR3\$\$S_RTFLG_012	= 00000003	TR4\$V_SCLASS_57	= 00000005
TR3\$\$S_TR3MSG	= 00000001	TR4\$V_SCLASS_BC	= 00000004
TR3\$V_MSG_CTL	= 00000000	TR4\$V_SCLASS_LS	= 00000002
TR3\$V_MSG_RTH	= 00000001	TR4\$V_SCLASS_METR	= 00000000
TR3\$V_RTFLG_012	= 00000000	TR4\$V_SCLASS_SUBA	= 00000003
TR3\$V_RTFLG_5	= 00000005	TRAN_DEVNAM	0000136C R 04
TR3\$V_RTFLG_7	= 00000007	UCB\$B_DEVCLASS	= 00000040
TR3\$V_RTFLG_PH2	= 00000006	UCB\$B_DEVTYPE	= 00000041
TR3\$V_RTFLG_RQR	= 00000003	UCB\$L_DEVDEPEND	= 00000044
TR3\$V_RTFLG_RTS	= 00000004	UCB\$W_DEVBUFSIZ	= 00000042
TR4\$\$S_QUAL_ADDR	= 00000000	X25_CNT_TAB	00000024 R 02
TR4\$\$S_QUAL_RTFLG	= 00000000	XMSV_CHR_HDPLX	= 00000002
TR4\$\$S_QUAL_SCLASS	= 00000000	XMSV_CHR_LOOPB	= 00000001
TR4\$C_BCE_MID1	= 040000AB	XMSV_CHR_MOP	= 00000000
TR4\$C_BCE_MID2	= 00000000	XMSV_STS_ACTIVE	= 0000000B
TR4\$C_BCR_MID1	= 030000AB	_SS_	= 000000EF
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	= 0000360		
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		

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

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	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
\$255\$DUA28:[SHRLIB]NMALIBRY.MLB;1	1
\$255\$DUA28:[SHRLIB]EVCDEF.MLB;1	1
\$255\$DUA28:[NETACP.OBJ]NETDRV.MLB;1	1
\$255\$DUA28:[NETACP.OBJ]NET.MLB;1	18
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	3
\$255\$DUA28:[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=LI\$:NETCNFDLL/OBJ=OBJ\$:NETCNFDLL MSRC\$:NETCNFDLL/UPDATE=(ENH\$:NETCNFDLL)+EXECMLS/LIB+LIB\$:NET/LIB+LIB\$:NETDRV/LIB+SHRLIB\$

0274 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

NETCNF
LIS

NETCNFACT
LIS

NETCNFDLL
LIS

0275 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

