

```
NNN      NNN  EEEEEEEEEEEEEEE TTTTTTTTTTTTTTTT  AAAAAAAAAA  CCCCCCCCCCCC  PPPPPPPPPPPP
NNN      NNN  EEEEEEEEEEEEEEE TTTTTTTTTTTTTTTT  AAAAAAAAAA  CCCCCCCCCCCC  PPPPPPPPPPPP
NNN      NNN  EEEEEEEEEEEEEEE TTTTTTTTTTTTTTTT  AAAAAAAAAA  CCCCCCCCCCCC  PPPPPPPPPPPP
NNN      NNN  EEE               TTT               AAA               AAA  CCC               PPP           PPP
NNN      NNN  EEE               TTT               AAA               AAA  CCC               FPP           PPP
NNN      NNN  EEE               TTT               AAA               AAA  CCC               PPP           PPP
NNNNNN   NNN  EEE               TTT               AAA               AAA  CCC               PPP           PPP
NNNNNN   NNN  EEE               TTT               AAA               AAA  CCC               PPP           PPP
NNNNNN   NNN  EEE               TTT               AAA               AAA  CCC               PPP           PPP
NNN      NNN  EEEEEEEEEEEEEEE TTT               AAA               AAA  CCC               PPPPPPPPPPPP
NNN      NNN  EEEEEEEEEEEEEEE TTT               AAA               AAA  CCC               PPPPPPPPPPPP
NNN      NNN  EEEEEEEEEEEEEEE TTT               AAA               AAA  CCC               PPPPPPPPPPPP
NNN      NNN  EEE               TTT               AAA               AAA  CCC               PPP
NNN      NNN  EEE               TTT               AAA               AAA  CCC               PPP
NNN      NNN  EEE               TTT               AAA               AAA  CCC               PPP
NNN      NNN  EEE               TTT               AAA               AAA  CCC               PPP
NNN      NNN  EEE               TTT               AAA               AAA  CCC               PPP
NNN      NNN  EEEEEEEEEEEEEEE TTT               AAA               AAA  CCCCCCCCCCCC  PPP
NNN      NNN  EEEEEEEEEEEEEEE TTT               AAA               AAA  CCCCCCCCCCCC  PPP
NNN      NNN  EEEEEEEEEEEEEEE TTT               AAA               AAA  CCCCCCCCCCCC  PPP
```

```

NN      NN      EEEEEEEEEEE TTTTTTTTTT DDDDDDDDD RRRRRRRR VV      VV      SSSSSSSS EEEEEEEEEEE SSSSSSSS
NN      NN      EEEEEEEEEEE TTTTTTTTTT DDDDDDDDD RRRRRRRR VV      VV      SSSSSSSS EEEEEEEEEEE SSSSSSSS
NN      NN      EE          TT          DD      DD  RR      RR  VV      VV  SS      SS      EEEEEEEEEEE SSSSSSSS
NN      NN      EE          TT          DD      DD  RR      RR  VV      VV  SS      SS      EEEEEEEEEEE SSSSSSSS
NNNN      NN      EE          TT          DD      DD  RR      RR  VV      VV  SS      SS      EEEEEEEEEEE SSSSSSSS
NNNN      NN      EE          TT          DD      DD  RR      RR  VV      VV  SS      SS      EEEEEEEEEEE SSSSSSSS
NN      NN      EEEEEEEEEEE TT          DD      DD  RRRRRRRR VV      VV  SSSSSS  EEEEEEEEEEE SSSSSS
NN      NN      EEEEEEEEEEE TT          DD      DD  RRRRRRRR VV      VV  SSSSSS  EEEEEEEEEEE SSSSSS
NN      NN      EE          TT          DD      DD  RR      RR  VV      VV  SS      SS      EEEEEEEEEEE SSSSSS
NN      NN      EE          TT          DD      DD  RR      RR  VV      VV  SS      SS      EEEEEEEEEEE SSSSSS
NN      NN      EE          TT          DD      DD  RR      RR  VV      VV  SS      SS      EEEEEEEEEEE SSSSSS
NN      NN      EE          TT          DD      DD  RR      RR  VV      VV  SS      SS      EEEEEEEEEEE SSSSSS
NN      NN      EEEEEEEEEEE TT          DDDDDDDDD RR      RR  VV      VV  SSSSSSSS EEEEEEEEEEE SSSSSSSS
NN      NN      EEEEEEEEEEE TT          DDDDDDDDD RR      RR  VV      VV  SSSSSSSS EEEEEEEEEEE SSSSSSSS

```

```

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

```

(2)	38	HISTORY
(4)	76	DECLARATIONS
(13)	352	FUNCTION DECISION TABLE
(14)	378	State Table
(27)	671	NET\$AZ_DR_TABLE - Disconnect Reason Code Mapping
(30)	805	NET\$FORK - Fork the XWB to do new work
(31)	851	NET\$SEND_EVENT - Abort current event without changing state
(31)	852	NET\$COMPLEX_EV - Change state and process new event
(31)	853	NET\$PRE_EMPTY - Process new event without changing state
(32)	908	NET\$EVENT - Event dispatcher
(33)	1009	NET\$SCH_MSG - schedule message transmission
(36)	1189	ACT\$NOP - Null action routine
(36)	1190	ACT\$BUG - BUG_CHECK action routine
(36)	1191	ACT\$LOG - Log-event action routine
(36)	1192	ACT\$NOLINK - Report 'SS\$ FILNOTACC'
(36)	1193	ACT\$SSABORT - Abort QIO since link was disconnected
(37)	1215	NET\$STARTIO - Start I/O operation
(38)	1295	NET\$FDT_SETMODE - Process IOS_SETMODE request
(39)	1322	NET\$FDT_CONTROL - IOS_ACPCONTROL FDT processing
(39)	1323	NET\$CONTROL - IOS_ACPCONTROL 'startio' processing
(40)	1417	NET\$FDT_ACCESS - IOS_ACCESS FDT processing
(40)	1418	NET\$ACCESS - IOS_ACCESS 'startio' processing
(41)	1518	ACT\$INITIATE - Connect Initiate action routine
(41)	1519	ACT\$CONFRIM - Connect Confirm action routine
(42)	1637	NET\$CMPL_ACC - Complete IOS_ACCESS, fill in window
(43)	1697	ACT\$ENT_RUN - Enter RUN state action routine
(44)	1718	NET\$FDT_DEACCESS - IOS_DEACCESS FDT processing
(44)	1719	NET\$DEACCESS - IOS_DEACCESS 'startio' processing
(47)	1872	CLEANUP_ACCESS - Cleanup XWB for terminated IOS_ACCESS
(48)	1928	NET\$CANCEL - Cancel I/O routine
(49)	1986	NET\$PURG_RUN - Cleanup XWB to exit RUN state
(50)	2151	NET\$ACP_COMM - Entry for ACP communication
(51)	2512	NET\$SEND_CS_MBX - Send counted string to mailbox
(52)	2568	NET\$SEND_MBX - Co-routine to send mailbox message
(53)	2656	NET\$CREATE_XWB - Create XWB for logical-link
(54)	2755	XWB_LOCLNK - Get XWB via local link number
(54)	2778	NET\$XWB_LOCLNK - Get XWB via local link number
(55)	2812	NET\$RET_SLOT - Return logical-link XWB slot if done
(55)	2813	NET\$QUE_XWB - Queue XWB to NETACP's AQB
(55)	2867	NET\$DRAIN_FREE_CXB - Drain CXB free queue
(56)	2883	NET\$ALONPGD_Z - Allocate and zero from system pool
(56)	2884	NET\$ALONONPAGED - Allocate from system pool
(57)	2931	NET\$DEALLOCATE - Deallocate non-paged pool
(58)	2957	NET\$MOV_TO_XWB - Move counted string to XWB\$B_DATA
(58)	2958	NET\$MOV_CSTR - Move counted string with count field
(58)	2959	NET\$MOV_USTR - Move counted string without count field
(59)	3003	NET\$POST_IO - Send IRP to COM\$POST

```
0000 1 .TITLE NETDRVSES - DECnet Session Control Module for NETDRIVER
0000 2 .IDENT 'V04-000'
0000 3 :
0000 4 :*****
0000 5 :
0000 6 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 :* ALL RIGHTS RESERVED.
0000 9 :*
0000 10 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 :* TRANSFERRED.
0000 16 :*
0000 17 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 :* CORPORATION.
0000 20 :*
0000 21 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 :*
0000 24 :*
0000 25 :*****
0000 26 :
0000 27 :++
0000 28 : FACILITY: DECNET
0000 29 :
0000 30 : ABSTRACT: This module is part of NETDRIVER and is the interface between
0000 31 : the user and the NSP layer.
0000 32 :
0000 33 : ENVIRONMENT: KERNEL mode, normal driver environment.
0000 34 :
0000 35 :--
0000 36 :
```

```
0000 38 .SBTTL HISTORY
0000 39
0000 40 :
0000 41 : AUTHOR: Alan D. Eldridge, CREATION DATE: 11-JUN-79
0000 42 :
0000 43 : MODIFIED BY:
0000 44 :
0000 45 : V03-022 LMP0308 L. Mark Pilant, 31-Aug-1984 16:15
0000 46 : Change default state of the ACL queue in the ORB.
0000 47 :
0000 48 : V03-021 ADE1042 A. Eldridge 23-Aug-1984
0000 49 : Don't create an XWB if the RCB$W_MCOUNT is zero. This condition
0000 50 : indicates that NETACP is going away and the test is needed to
0000 51 : avoid a race condition that can crash the system.
0000 52 :
0000 53 : V03-020 ADE1041 A. Eldridge 25-Jun-1984
0000 54 : Fix loop problem in cleaning up receives. Return SS$ CONNECFail
0000 55 : when an IO$ ACCESS fct can't locate the XWB (was SS$ NOLINKS).
0000 56 : Send NET$C DR ABORT upon IO$ DEACCESS!IO$M_ABORT (was sending
0000 57 : NET$C_DR_NORMAl).
0000 58 :
0000 59 : V03-019 LMP0221 L. Mark Pilant, 7-Apr-1984 14:29
0000 60 : Change UCB$$_OWNUIC to ORB$$_OWNER and UCB$$_VPROT to
0000 61 : ORB$$_PROT.
0000 62 :
0000 63 : V03-018 ADE1041 A. Eldridge 7-Mar-1984
0000 64 : Fix resource error count -- registers were screwed up.
0000 65 :
0000 66 : V03-017 ADE1040 A. Eldridge 10-Sep-1983
0000 67 : Major rewrite to accomodate changes to allow NSP (NETDRVNSP.MAR)
0000 68 : to use kernel mode AST's to nibble away at the user buffers
0000 69 : rather than accessing them just at FDT or I/O post time. This
0000 70 : change was needed to allow huge user buffers (for performance)
0000 71 : without requiring a lot of pool.
0000 72 :
0000 73 :
```

```

0000 75
0000 76 .SBTTL DECLARATIONS
0000 77
0000 78 : INCLUDE FILES:
0000 79 :
0000 80
0000 81 $AQBDEF
0000 82 $ACBDEF
0000 83 $CCBDEF
0000 84 $CRBDEF
0000 85 $CXBDEF
0000 86 $DDBDEF
0000 87 $DRDEF
0000 88 $DYNDEF
0000 89 $FKBDEF
0000 90 $IODEF
0000 91 $IPLDEF
0000 92 $IRPDEF
0000 93 $JIBDEF
0000 94 $MSGDEF
0000 95 $ORBDEF
0000 96 $PCBDEF
0000 97 $PHDDEF
0000 98 $PRDEF
0000 99 $SSDEF
0000 100 $TQDEF
0000 101 $UCBDEF
0000 102 $VECDEF
0000 103
0000 104 $ICBDEF
0000 105 $IDBDEF
0000 106 $LPDDEF
0000 107 $LTBDEF
0000 108 $LLIDEF
0000 109 $RCBDEF
0000 110
0000 111 $NETSYMDEF
0000 112 $NETUPDDEF
0000 113 $NSPMSGDEF
0000 114
0000 115 $CXBEXTDEF ; NETDRIVER CXB extensions
0000 116 $XWBDEF ; XWB and LSB definitions
0000 117

```

```
0000 119
0000 120 :
0000 121 : MACROS:
0000 122 :
0000 123 :
0000 124 :
0000 125 : Bit definition macro
0000 126 :
0000 127 .MACRO BITDEF BLK,SYM,BITVAL
0000 128
0000 129         'BLK'$V 'SYM' = BITVAL
0000 130         'BLK'$M 'SYM' = 1<BITVAL>
0000 131 .ENDM
0000 132 :
0000 133 :
0000 134 : Macro to set up mailbox message filtering table
0000 135 :
0000 136 .MACRO MBX_FILTER MESSAGE,BIT
0000 137
0000 138         .LONG MBX$M 'BIT
0000 139         .WORD MSG$M 'MESSAGE
0000 140
0000 141 .ENDM MBX_FILTER
0000 142 :
0000 143 :
0000 144 : Macro to build a mask of XWB$M_FLG_xxx bits
0000 145 :
0000 146 .MACRO BLDMSK A
0000 147         _$MSK = _$MSK + XWB$M_FLG_'A'
0000 148 .ENDM
0000 149 :
0000 150 :
0000 151 : Macro to fill the 'set' and 'clear' XWB$W_FLG tables
0000 152 :
0000 153 .MACRO STATEMASK STA,SETM,CLRM
0000 154
0000 155 :
0000 156 : Build and enter the 'set FLG' bit mask
0000 157 :
0000 158         _$MSK = 0
0000 159         .IRP A,<SETM>
0000 160             BLDMSK A
0000 161         .ENDR
0000 162         . = NET$AW_FLG_SETM + <2*XWB$C_STA_'STA'>
0000 163         .WORD _$MSK
0000 164 :
0000 165 : Build and enter the 'clear FLG' bit mask
0000 166 :
0000 167         _$MSK = 0
0000 168         .IRP A,<CLRM>
0000 169             BLDMSK A
0000 170         .ENDR
0000 171         . = NET$AW_FLG_CLRM + <2*XWB$C_STA_'STA'>
0000 172         .WORD _$MSK
0000 173 .ENDM
0000 174
```

```
0000 176
0000 177 :
0000 178 : Macro to initialize NSP state tables
0000 179 :
0000 180 .MACRO STTAB ; Init state transition data
0000 181
0000 182     _$EVENT_INDEX = 0 ; Init event index
0000 183     _$ACT_INDEX = 0 ; Init action routine index
0000 184     ACT$_BUG == 0 ; Init the "bug-check" action routine
0000 185 ; index
0000 186
0000 187     _$ACT_DFLT = <XWB$C_STA_CLO - ; Default state table entry
0000 188                 @NET$C_ACTBITS> - ;
0000 189                 + ACT$_BUG ;
0000 190
0000 191
0000 192 NET$AW_FLG_SETM: ; Bits to be set upon entering state
0000 193                 .BLKW XWB$C_NUMSTA ; and upon timeout
0000 194 NET$AW_FLG_CLRM: .BLKW XWB$C_NUMSTA ; Bits to be cleared upon entering state
0000 195
0000 196 NET$AB_STTAB: .BLKB 0 ; Bind the table address
0000 197
0000 198 .ENDM
0000 199
0000 200 :
0000 201 : Macro to move the current position within the state table
0000 202 :
0000 203 .MACRO ENDSTTAB ; Move PC to end of table
0000 204     . = NET$AB_STTAB -
0000 205         + <_SEVENT_INDEX * XWB$C_NUMSTA>
0000 206 .ENDM
0000 207
0000 208
0000 209 .MACRO EVENT EV ; Setup for this event
0000 210
0000 211     EV == _SEVENT_INDEX ; Define event code
0000 212     . = NET$AB_STTAB + <EV * XWB$C_NUMSTA> ; Move PC to proper event
0000 213     .BYTE <_SACT_DFLT>[XWB$C_NUMSTA] ; Init the entry
0000 214
0000 215     . = NET$AB_STTAB + <EV * XWB$C_NUMSTA> ; Move PC to proper event
0000 216     _SEVENT_INDEX = _SEVENT_INDEX + 1 ; Get ready for next event
0000 217
0000 218 .ENDM
0000 219
0000 220
0000 221 :
0000 222 : Macro to fill the build and enter the state transition table element
0000 223 :
0000 224 .MACRO STATE CURSTA,NXTSTA,ACTION,?LL ; Make table entry
0000 225     LL:
0000 226         . = .+XWB$C_STA_'CURSTA' ; Goto state table entry
0000 227
0000 228
0000 229     ; If the action routine index is not defined then define the index
0000 230
0000 231     ; Create the state-table entry.
0000 232
```

```

0000 233      :
0000 234      :IF NDF,ACTS_'ACTION'
0000 235      ACTS_'ACTION' = _SACT_INDEX
0000 236      _SACT_INDEX = _SACT_INDEX + 1
0000 237      .ENDC
0000 238      .BYTE <XWBSCTA_'NXTSTA' @NETSC_ACTBITS> + ACTS_'ACTION'
0000 239
0000 240      .=LL
0000 241      .ENDM
0000 242

```

```
0000 244
0000 245 :
0000 246 : EQUATED SYMBOLS:
0000 247 :
0000 248 :
0000 249 :
0000 250 : Argument list offsets for QIO
0000 251 :
00000000 0000 252 P1 = 0 ; Buffer address
00000004 0000 253 P2 = 4 ; Buffer length
00000008 0000 254 P3 = 8 ; Miscellaneous
0000 255
0000 256
0000 257 ASSUME FKB$C_LENGTH LE ACB$C_LENGTH
0000 258
00000120 0000 259 $tmp == <XWB$$XWB+7>&^C<7> ; XWB length, quad word aligned
00000160 0000 260 XWB$$ == $tmp+64 ; Allow enough room for the route-header
00000120 0000 261 XWB$$_PTR_RTHD == $tmp ; Ptr to route-header
00000124 0000 262 XWB$$_ADJ_INX == $tmp+4 ; Adjacency index
00000158 0000 263 XWB$$_TR3HDR == XWB$$-8 ; Start of standard Phase III header
0000 264 ; (must be quadword aligned)
0000 265
0000017C 0000 266 XWB_C_LEN = XWB$$+ACB$C_LENGTH ; Total XWB length
0000 267
0000 268 :
0000 269 : Definitions for mailbox message filtering
0000 270 :
0000 271 $VIELD MBX,0,<-
0000 272 <NETSTATE,,M>,- ; Network state change
0000 273 <EVTAVL,,M>,- ; Events available for logging
0000 274 <EVTRCVCHG,,M>,- ; Event receiver database change
0000 275 <EVTXMTCHG,,M>,- ; Event xmitter database change
0000 276 >
0000 277
0000 278 :
0000 279 :
0000 280 : Define a mask containing all bits indicating work needs to be done
0000 281 :
0000 282 :
0000 283 XWB$M_FLG_WMSK = XWB$M_FLG_SCD ! -
0000 284 XWB$M_FLG_SDT ! XWB$M_FLG_SDACK!-
0000 285 XWB$M_FLG_SLI ! XWB$M_FLG_SIACK!-
0000039D 0000 286 XWB$M_FLG_CLO ! XWB$M_FLG_BREAK
0000 287
```

```
0000 289
0000 290 : DRIVER PROLOGUE TABLE
0000 291 :
0000 292 :
0000 293 :
0000 294
0000 295 DPTAB .PSECT $$$105_PROLOGUE
0000 296 END = NETSEND,- : Define driver prologue table
0000 297 ADAPTER = NULL,- : End of driver
0000 298 UCBSIZE = UCBS$ LENGTH,- : Adapter type
0000 299 NAME = NETDRIVER : UCB size
0038 300 : Driver name
0038 301 DPT_STORE INIT : CONTROL BLOCK INIT VALUES
0038 302
0038 303 DPT_STORE CRB,CRB$ INTD+VEC$ ADP,L,0 : No ADP pointer
003F 304 DPT_STORE UCB,UCB$ MB SEED,W,0 : Init. unit value for cloning
0044 305 DPT_STORE UCB,UCB$ FIPL,B,NET$C IPL : Fork IPL
0048 306 DPT_STORE UCB,UCB$ DIPL,B,NET$C IPL : Device IPL
004C 307 DPT_STORE ORB,ORB$ FLAGS,B,-
004C 308 <ORB$ PROT_16> : SOGW protection word
005C 309 DPT_STORE ORB,ORB$ PROT,W,0 : Default protection
0055 310 DPT_STORE ORB,ORB$ OWNER,L,<^X010001> : Owner UIC
005C 311 DPT_STORE UCB,UCB$ DEVCHAR,L,- : Device characteristics
005C 312 <DEV$M NET!- : Network device
005C 313 DEV$M AVL!- : Available
005C 314 DEV$M MBX!- : Mailbox type (no hardware)
005C 315 DEV$M IDV!- : Input device
005C 316 DEV$M ODV- : Output device
005C 317 >
0063 318 DPT_STORE UCB,UCB$ DEVBUFSIZ,W,256
0068 319 DPT_STORE UCB,UCB$ DEVDEPEND,L,-
0068 320 MBX$M NETSTATE : Enable NETSHUT by default
006F 321 DPT_STORE UCB,UCB$ STS,W,-
006F 322 <UCB$M ONLINE!- : Device online
006F 323 UCB$M TEMPLATE- : NET0 is the "template" UCB
006F 324 > : used to build other NET UCBs
0074 325
0074 326 DPT_STORE REINIT : CONTROL BLOCK RE-INIT VALUES
0074 327
0074 328 DPT_STORE DDB,DDB$ DDT,D,NET$DDT
0079 329 DPT_STORE CRB,CRB$ INTD+VEC$ INITIAL,D,NET$CTLR_INIT
007E 330 DPT_STORE CRB,CRB$ INTD+VEC$ UNITINIT,D,NET$UNIT_INIT
0083 331 DPT_STORE CRB,CRB$ INTD+VEC$ START,D,NET$ACP COMM
0088 332 DPT_STORE CRB,CRB$ INTD+4,D,NET$INTERRUPT
008D 333
008D 334 DPT_STORE END :
0000 335
0000 336
```

```
0000 338
0000 339 :
0000 340 : DRIVER DISPATCH TABLE
0000 341 :
00000000 342 .PSECT $$$115_DRIVER, LONG
0000 343
0000 344 DDTAB DEVNAM = NET, - ; DRIVER DISPATCH TABLE
0000 345 FUNCTB = FUNCTABLE, - ; Function decision table address
0000 346 START = NET$STARTIO, - ; Start I/O operation
0000 347 ALTSTART = NET$ALTENTRY, - ; Alternate I/O request entry point
0000 348 CANCEL = NET$CANCEL, - ; Cancel I/O entry point
0000 349 UNSOLIC = NET$UNSOL_INTR; Unsolicited interrupt
0038 350
0038 351
0038 352 .SBTTL FUNCTION DECISION TABLE
0038 353
0038 354 FUNCTABLE: ; FUNCTION DECISION TABLE
0038 355 FUNCTAB - ; Legal Functions
0038 356 <READVBLK, READLBLK, - ; Read
0038 357 WRITEVBLK, WRITELBLK, - ; Write
0038 358 SETMODE, - ; Set mailbox message filters
0038 359 ACCESS, - ; Logical-link Connect/Reject
0038 360 ACPCONTROL, - ; ACP Control function
0038 361 DEACCESS, - ; Logical-link Disconnect
0038 362 >
0040 363 FUNCTAB - ; BUFFERED I/O FUNCTIONS
0040 364 <READVBLK, READLBLK, - ; Read
0040 365 WRITEVBLK, WRITELBLK, - ; Write
0040 366 SETMODE, - ; Set mailbox message filters
0040 367 ACCESS, - ; Logical-link Connect/Reject
0040 368 ACPCONTROL, - ; ACP Control function
0040 369 DEACCESS, - ; Logical-link Disconnect
0040 370 >
0048 371 FUNCTAB NET$FDT_RCV, <READLBLK> ; Read
0054 372 FUNCTAB NET$FDT_XMT, <WRITELBLK> ; Write
0060 373 FUNCTAB NET$FDT_ACCESS, <ACCESS> ; Connect Logical-link
006C 374 FUNCTAB NET$FDT_DEACCESS, <DEACCESS> ; Disconnect Logical-link
0078 375 FUNCTAB NET$FDT_SETMODE, <SETMODE> ; Set mailbox message filters
0084 376 FUNCTAB NET$FDT_CONTROL, <ACPCONTROL> ; ACP Control
```

```
0090 378 .SBTTL State Table
0090 379
0090 380 :
0090 381 : OWN STORAGE:
0090 382 :
00000080 0090 383 PATCH_AREA_SIZE = 128 ; Size of patch area space
0090 384
00000080 0090 385 NET$GQ_PATCH::
00000098' 0090 386 .LONG PATCH_AREA_SIZE
0098 387 .LONG .+4 ; (not an address - offset from start
0098 388 ; of image to base of patch space)
00000118 0098 389 .BLKB PATCH_AREA_SIZE
0118 390
0118 391
FFFFFEF5' 0118 392 NET$GL_OFF_DPTFLG:: .LONG DPT$TAB + DPT$B_FLAGS - . ; Offset to DPT$B_FLAGS
011C 393
00000005 011C 394 NET$C_ACTBITS = 5 ; Number of action bits per entry
00000003 011C 395 NET$C_STABITS = 3 ; Number of state bits per entry
000000E0 011C 396 NET$M_STAMSK = <7>a5 ; State bit mask
011C 397
011C 398 :
011C 399 :
011C 400 : The following definitions must be contiguous to the NSPTABLES definition
011C 401 :
011C 402 :
011C 403 STTAB ; Init state transition table
013C 404
013C 405 EVENT NETEVTS CI ; CI message received
013C 406 STATE CIS, CIS, BUG ; something wrong in the driver
013C 407 STATE CAR, CAR, LOG ; unexpected event
013C 408 STATE CIR, CIR, RCV_CR ; Assume received retransmitted CI
013C 409 STATE CCS, CCS, RCV_CR ; Assume received retransmitted CI
013C 410 STATE RUN, RUN, LOG ; unexpected event
013C 411 STATE DIS, DIS, LOG ; unexpected event
013C 412 STATE DIR, DIR, LOG ; unexpected event
013C 413 STATE CLO, CIR, RCV_CI ; inbound connect sequence
013C 414
013C 415 EVENT NETEVTS CA ; Connect Ack received
0144 416 STATE CIS, CAR, RCV_CA ; measure initial round-trip time
0144 417 STATE CAR, CAR, NOP ; assume retransmission
0144 418 STATE CIR, CIR, LOG ; unexpected event
0144 419 STATE CCS, CCS, LOG ; unexpected event
0144 420 STATE RUN, RUN, NOP ; assume late arrival
0144 421 STATE DIS, DIS, NOP ; assume late arrival
0144 422 STATE DIR, DIR, NOP ; assume late arrival
0144 423 STATE CLO, CLO, NOP ; assume late arrival
0144 424
0144 425 EVENT NETEVTS CC ; Connect Confirm received
014C 426 STATE CIS, RUN, RCV_CC ; normal handshaking sequence
014C 427 STATE CAR, RUN, RCV_CC ; normal handshaking sequence
014C 428 STATE CIR, CIR, LOG ; unexpected event
014C 429 STATE CCS, CCS, LOG ; unexpected event
014C 430 STATE RUN, RUN, NOP ; assume retransmission
014C 431 STATE DIS, DIS, NOP ; we enter DIS for many reasons
014C 432 STATE DIR, DIR, NOP ; assume late arrival
014C 433 STATE CLO, CLO, RTS_NLT ; assume late arrival
014C 434
```

014C	436					
014C	437	EVENT	NETEVTS PH2CCS			; Phase II connect confirm xmt-complete
0154	438	STATE	CIS, CIS,	NOP		
0154	439	STATE	CAR, CAR,	NOP		
0154	440	STATE	CIR, CIR,	NOP		
0154	441	STATE	CCS, RUN,	ENT_RUN		; Normal Phase II handshaking sequence
0154	442	STATE	RUN, RUN,	NOP		
0154	443	STATE	DIS, DIS,	NOP		
0154	444	STATE	DIR, DIR,	NOP		
0154	445	STATE	CLO, CLO,	NOP		
0154	446					
0154	447	EVENT	NETEVTS RTS			; Rcv 'return to sender' CI message
015C	448	STATE	CIS, CLO,	RCV_RTS		; Process returned message
015C	449	STATE	CAR, CAR,	NOP		; Assume late arrival on retransmission
015C	450	STATE	CIR, CIR,	NOP		; Assume late arrival on retransmission
015C	451	STATE	CCS, CCS,	NOP		; Assume late arrival on retransmission
015C	452	STATE	RUN, RUN,	NOP		; Assume late arrival on retransmission
015C	453	STATE	DIS, DIS,	NOP		; Assume late arrival on retransmission
015C	454	STATE	DIR, DIR,	NOP		; Assume late arrival on retransmission
015C	455	STATE	CLO, CIR,	NOP		; Assume late arrival on retransmission
015C	456					
015C	457	EVENT	NETEVTS DATA			; Data message received
0164	458	STATE	CIS, CIS,	LOG		; unexpected event
0164	459	STATE	CAR, CAR,	LOG		; unexpected event
0164	460	STATE	CIR, CIR,	LOG		; unexpected event
0164	461	STATE	CCS, RUN,	ENT_RUN		; a normal handshaking sequence
0164	462	STATE	RUN, RUN,	RCV_DATA		; this is what NSP is for
0164	463	STATE	DIS, DIS,	NOP		; unavoidable race in sending DI
0164	464	STATE	DIR, DIR,	NOP		; assume late arrival
0164	465	STATE	CLO, CLO,	RTS_NLT		; assume late arrival
0164	466					
0164	467	EVENT	NETEVTS DTACK			; Data Ack received
016C	468	STATE	CIS, CIS,	LOG		; unexpected event
016C	469	STATE	CAR, CAR,	LOG		; unexpected event
016C	470	STATE	CIR, CIR,	LOG		; unexpected event
016C	471	STATE	CCS, RUN,	ENT_RUN		; a normal handshaking sequence
016C	472	STATE	RUN, RUN,	RCV_DTACK		; drive the link
016C	473	STATE	DIS, DIS,	NOP		; assume late arrival or race
016C	474	STATE	DIR, DIR,	NOP		; assume late arrival or race
016C	475	STATE	CLO, CLO,	RTS_NLT		; assume late arrival or race
016C	476					

016C	478					
016C	479	EVENT	NETEVTS LS			: Link Service msg received
0174	480	STATE	CIS, CIS,	LOG		: unexpected event
0174	481	STATE	CAR, CAR,	LOG		: unexpected event
0174	482	STATE	CIR, CIR,	LOG		: unexpected event
0174	483	STATE	CCS, RUN,	ENT_RUN		: a normal handshaking sequence
0174	484	STATE	RUN, RUN,	RCV_LI		: drive the link
0174	485	STATE	DIS, DIS,	NOP		: assume late arrival or race
0174	486	STATE	DIR, DIR,	NOP		: assume late arrival or race
0174	487	STATE	CLO, CLO,	RTS_NLT		: assume late arrival or race
0174	488					
0174	489	EVENT	NETEVTS INT			: Interrupt msg received
017C	490	STATE	CIS, CIS,	LOG		: unexpected event
017C	491	STATE	CAR, CAR,	LOG		: unexpected event
017C	492	STATE	CIR, CIR,	LOG		: unexpected event
017C	493	STATE	CCS, RUN,	ENT_RUN		: a normal handshaking sequence
017C	494	STATE	RUN, RUN,	RCV_LI		: drive the link
017C	495	STATE	DIS, DIS,	NOP		: assume late arrival or race
017C	496	STATE	DIR, DIR,	NOP		: assume late arrival or race
017C	497	STATE	CLO, CLO,	RTS_NLT		: assume late arrival or race
017C	498					
017C	499	EVENT	NETEVTS LIACK			: INT/LS Ack received
0184	500	STATE	CIS, CIS,	LOG		: unexpected event
0184	501	STATE	CAR, CAR,	LOG		: unexpected event
0184	502	STATE	CIR, CIR,	LOG		: unexpected event
0184	503	STATE	CCS, RUN,	ENT_RUN		: a normal handshaking sequence
0184	504	STATE	RUN, RUN,	RCV_LIACK		: drive the link
0184	505	STATE	DIS, DIS,	NOP		: assume late arrival or race
0184	506	STATE	DIR, DIR,	NOP		: assume late arrival or race
0184	507	STATE	CLO, CLO,	RTS_NLT		: assume late arrival or race
0184	508					
0184	509	EVENT	NETEVTS DI			: Disconnect Initiate msg rcv'd
018C	510	STATE	CIS, DIR,	RCV_Dx		: link rejected
018C	511	STATE	CAR, DIR,	RCV_Dx		: link rejected
018C	512	STATE	CIR, DIR,	ABORT		: abort the link, no local owner
018C	513	STATE	CCS, DIR,	RCV_Dx		: abort the link
018C	514	STATE	RUN, DIR,	RCV_Dx		: abort the link
018C	515	STATE	DIS, DIR,	ABORT		: change state and send DC
018C	516	STATE	DIR, DIR,	NOP		: send DC
018C	517	STATE	CLO, CLO,	RTS_NLT		: assume race or late arrival
018C	518					
018C	519	EVENT	NETEVTS DC			: Disconnect Confirm msg rcv'd
0194	520	STATE	CIS, CLO,	RCV_Dx		: link rejected
0194	521	STATE	CAR, CLO,	RCV_Dx		: link rejected
0194	522	STATE	CIR, CLO,	ABORT		: link aborted, no local owner
0194	523	STATE	CCS, CLO,	RCV_Dx		: link aborted
0194	524	STATE	RUN, CLO,	RCV_Dx		: link aborted
0194	525	STATE	DIS, CLO,	NOP		: normal handshaking sequence
0194	526	STATE	DIR, CLO,	NOP		: assume DC is a 'no link terminate'
0194	527	STATE	CLO, CLO,	NOP		: assume late arrival
0194	528					
0194	529					

0194	531					
0194	532	EVENT	NETEVTS	DSCLNK		; Link failed confidence test
019C	533	STATE	CIS, CLO,	ABORT		; connect timed out
019C	534	STATE	CAR, DIS,	ABORT		; connect timed out
019C	535	STATE	CIR, DIS,	ABORT		; local system is slow
019C	536	STATE	CCS, DIS,	ABORT		; connect timed out
019C	537	STATE	RUN, DIS,	ABORT		; problem talking with remote node
019C	538	STATE	DIS, CLO,	NOP		; abort the link
019C	539	STATE	DIR, CLO,	NOP		; abort the link
019C	540	STATE	CLO, CLO,	NOP		; Try to deallocate XWB
019C	541					
019C	542					
019C	543	EVENT	NETEVTS	CANLNK		; Local cancel of link
01A4	544	STATE	CIS, CLO,	CANLNK		; abort from a Connect state
01A4	545	STATE	CAR, CLO,	CANLNK		; abort from a Connect state
01A4	546	STATE	CIR, DIS,	CANLNK		; abort link, no local owner
01A4	547	STATE	CCS, DIS,	CANLNK		; abort from a Connect state
01A4	548	STATE	RUN, DIS,	CANLNK		; orderly shutdown
01A4	549	STATE	DIS, DIS,	NOP		; link is already disconnecting
01A4	550	STATE	DIR, DIR,	NOP		; link is already disconnecting
01A4	551	STATE	CLO, CLO,	NOP		; link is already disconnecting
01A4	552					
01A4	553	EVENT	NETEVTS	RESDIS		; Resume disconnect
01AC	554	STATE	CIS, CIS,	BUG		; Valid only from RUN state
01AC	555	STATE	CAR, CAR,	BUG		; Valid only from RUN state
01AC	556	STATE	CIR, CIR,	BUG		; Valid only from RUN state
01AC	557	STATE	CCS, CCS,	BUG		; Valid only from RUN state
01AC	558	STATE	RUN, DIS,	RES_DISC		; Disconnect if XWB is idle
01AC	559	STATE	DIS, DIS,	BUG		; Valid only from RUN state
01AC	560	STATE	DIR, DIR,	BUG		; Valid only from RUN state
01AC	561	STATE	CLO, CLO,	BUG		; Valid only from RUN state
01AC	562					
01AC	563					

01AC	565					
01AC	566	EVENT	NETEVTS CIA			: Connect Initiate IOS_ACCESS
01B4	567	STATE	CIS, CIS,	BUG		: XWB was just created
01B4	568	STATE	CAR, CAR,	BUG		: XWB was just created
01B4	569	STATE	CIR, CAR,	BUG		: XWB was just created
01B4	570	STATE	CCS, CCS,	BUG		: XWB was just created
01B4	571	STATE	RUN, RUN,	BUG		: XWB was just created
01B4	572	STATE	DIS, DIS,	BUG		: XWB was just created
01B4	573	STATE	DIR, DIR,	BUG		: XWB was just created
01B4	574	STATE	CLO, CIS	INITIATE		: Normal connect initiate seq.
01B4	575					
01B4	576	EVENT	NETEVTS CCA			: Connect Confirm IOS_ACCESS
01BC	577	STATE	CIS, CIS,	SSABORT		: Confirm not possible
01BC	578	STATE	CAR, CAR,	SSABORT		: Confirm not possible
01BC	579	STATE	CIR, CCS,	CONFIRM		: Normal connect confirm seq.
01BC	580	STATE	CCS, CCS,	SSABORT		: Confirm not possible
01BC	581	STATE	RUN, RUN,	SHRLNK		: Second accessor to link
01BC	582	STATE	DIS, DIS,	SSABORT		: Confirm no longer possible
01BC	583	STATE	DIR, DIR,	SSABORT		: Confirm no longer possible
01BC	584	STATE	CLO, CLO,	SSABORT		: Confirm no longer possible
01BC	585					
01BC	586	EVENT	NETEVTS CRA			: Connect Reject IOS_ACCESS
01C4	587	STATE	CIS, CIS,	SSABORT		: Reject not possible
01C4	588	STATE	CAR, CAR,	SSABORT		: Reject not possible
01C4	589	STATE	CIR, DIS,	CONFIRM		: Normal connect reject seq.
01C4	590	STATE	CCS, CCS,	SSABORT		: Reject not possible
01C4	591	STATE	RUN, RUN,	SSABORT		: Reject not possible
01C4	592	STATE	DIS, DIS,	SSABORT		: Reject not possible
01C4	593	STATE	DIR, DIR,	SSABORT		: Reject not possible
01C4	594	STATE	CLO, CLO,	SSABORT		: Reject not possible
01C4	595					
01C4	596	EVENT	NETEVTS DEA			: QIO IOS_DEACCESS
01CC	597	STATE	CIS, CIS,	BUG		: Channel should not have window
01CC	598	STATE	CAR, CAR,	BUG		: Channel should not have window
01CC	599	STATE	CIR, CIR,	BUG		: Channel should not have window
01CC	600	STATE	CCS, CCS,	BUG		: Channel should not have window
01CC	601	STATE	RUN, DIS,	DEACCESS		: But change to DIS state only
01CC	602					: if this is the last accessor
01CC	603	STATE	DIS, DIS,	DEACCESS		: Link was aborted externally
01CC	604	STATE	DIR, DIR,	DEACCESS		: Link was aborted externally
01CC	605	STATE	CLO, CLO,	DEACCESS		: Link was aborted externally
01CC	606					

01CC	608					
01CC	609	EVENT	NETEVTS	MBXERR		; Fatal error writing to mailbox
01D4	610	STATE	CIS, CLO,	ABORT		; abort from a Connect state
01D4	611	STATE	CAR, CLO,	ABORT		; abort from a Connect state
01D4	612	STATE	CIR, DIS,	ABORT		; abort link, no local owner
01D4	613	STATE	CCS, DIS,	ABORT		; abort from a Connect state
01D4	614	STATE	RUN, DIS,	ABORT		; abort from the RUN state
01D4	615	STATE	DIS, DIS,	NOP		; link is already disconnecting
01D4	616	STATE	DIR, DIR,	NOP		; link is already disconnecting
01D4	617	STATE	CLO, CLO,	NOP		; link is already disconnecting
01D4	618					
01D4	619	EVENT	NETEVTS	PROERR		; Protocol error (NOP for now)
01DC	620	STATE	CIS, CIS,	NOP		:
01DC	621	STATE	CAR, CAR,	NOP		:
01DC	622	STATE	CIR, DIS,	NOP		:
01DC	623	STATE	CCS, DIS,	NOP		:
01DC	624	STATE	RUN, RUN,	NOP		:
01DC	625	STATE	DIS, DIS,	NOP		:
01DC	626	STATE	DIR, DIR,	NOP		:
01DC	627	STATE	CLO, CLO,	NOP		:
01DC	628					:
01DC	629					:

```
01DC 631
01DC 632 :
01DC 633 : Setup tables which specify which XWBSW_FLG bits to set and clear upon
01DC 634 : a transition into a new state.
01DC 635 :
01DC 636 : New
01DC 637 : State  Flags to set  Flags to clear
01DC 638 : -----
01DC 639 :
01DC 640 STATEMASK CIS, <SCD> <WBUF>
0130 641 STATEMASK CAR, <CLO> <WBUF>
0132 642 STATEMASK CIR, <SCD> <WBUF>
0134 643 STATEMASK CCS, <SCD> <WBUF>
0136 644 STATEMASK RUN, <SDT,SDFL,WHGL> <WBUF,SCD>
0138 645 STATEMASK DIR, <SCD> <WBUF,WBP,WHGL,WDAT,SDT,SLI,SDACK,SIACK,-
0138 646 : BREAK,IAVL,TBPR,SIFL,SDFL>
013A 647 STATEMASK DIS, <SCD> <WBUF,WBP,WHGL,WDAT,SDT,SLI,SDACK,SIACK,-
013A 648 : BREAK,IAVL,TBPR,SIFL,SDFL>
013C 649 STATEMASK CLO, <CLO> <WBUF,WBP,WHGL,WDAT,SDT,SLI,SDACK,SIACK,-
013C 650 : SCD,BREAK,IAVL,TBPR,SIFL,SDFL>
012E 651
012E 652 ENDSTTAB
01E4 653
01E4 654
01E4 655 :
01E4 656 : The following mask is used to identify the subset of flags used
01E4 657 : to signal work to be done
01E4 658 :
0000039D 01E4 659 NET$GL_WORKBITS:: .LONG XWBSM_FLG_WMSK ; Flags requiring work to be done
01E8 660
01E8 661
01E8 662 MBX_TABLE: ; Table for mapping mbx msg codes
01E8 663 : to filter bits
01E8 664 MBX_FILTER NETSHUT,NETSTATE ; Network state change
01EE 665 MBX_FILTER EVTAVL,EVTAVL ; Events available for logging
01F4 666 MBX_FILTER EVTRCVCHG,EVTRCVCHG ; Event receiver database change
01FA 667 MBX_FILTER EVTXTMCHG,EVTXTMCHG ; Event xmitter database change
00000000 0200 668 .LONG 0 ; End of table
0204 669
```

```
0204 671 .SBTTL NET$AZ_DR_TABLE - Disconnect Reason Code Mapping
0204 672
0204 673
0204 674 ;
0204 675 ; Macro to set up connect reject reason codes
0204 676 ;
00000000 0204 677 REASON_W_DR == 0 ; Reason code
00000002 0204 678 REASON_W_SS == 2 ; SS$... to return in data IRP's
00000004 0204 679 REASON_W_MBX == 4 ; MBX$... message code
00000006 0204 680 REASON_C_LENGTH == 6
0204 681
0204 682 .MACRO MRC REASON,SS_CODE,MSG_CODE
0204 683
0204 684 .WORD NET$C_DR 'REASON
0204 685 .WORD SS$ 'SS_CODE
0204 686 .WORD MSG$ 'MSG_CODE
0204 687
0204 688 .ENDM MRC
0204 689
00000064 0204 691 NET$C_DR_INVALID == 100 ; Fake value meaning 'not setup'
00000066 0204 692 NET$C_DR_DEACC == 102 ; Fake value for code conversion
0204 693
0204 694 NET$AZ_DR_TABLE: ; Table for mapping disconnect reasons.
0204 695 ; for other than the 'connect-initiate'
0204 696 ; state
0204 697 ;
0204 698 ;
0204 699 ; discon data mailbox
0204 700 ; reason status message
0204 701 ; -----
0204 702 ;
0204 703 MRC NORMAL, LINKDISCON, DISCON ;
0204 704 MRC EXIT, LINKEXIT, EXIT ; User exit or timeout
0210 705 MRC NOPATH, PATHLOST, PATHLOST ; Path lost to partner node
0216 706 MRC SHUT, SHUT, NETSHUT ; Node shutting down
021C 707 MRC NOBJ, PROTOCOL, ABORT ; No such object
0222 708 MRC ABORT, LINKABORT, ABORT ; Disconnect abort
0228 709 MRC THIRD, THIRDPARTY, THIRDPARTY ; Disconnect by third party
022E 710 MRC ACCESS, PROTOCOL, ABORT ; Login info invalid
0234 711 MRC RSU, PROTOCOL, ABORT ; Resource error
023A 712 MRC BUSY, PROTOCOL, ABORT ; Object too busy
0240 713 MRC FMT, PROTOCOL, ABORT ; Illegal process name field
0246 714 MRC NONODE, PROTOCOL, ABORT ; Unrecognized node i.d.
024C 715 MRC IVNODE, PROTOCOL, ABORT ; Invalid node-i.d. format
0252 716 ;
0252 717 ;
0252 718 ; The following are internal codes and are not part of NSP
0252 719 ;
0252 720 ;
0252 721 MRC DEACC, LINKDISCON, DISCON ; Link is IO$ DEACCESS'ed
0258 722 MRC INVALID, LINKABORT, ABORT ; Reason field never setup
025E 723 ;
FFFFF000 025E 724 .LONG -1 ; Terminate the table (the last entry
0262 725 ; is to be used as a catch-all)
0262 726
0262 727
```

```
0262 728
0262 729 NET$AZ_DR_CONTAB: ; Table for mapping reject reasons
0262 730 ; in one of the 'connect' states
0262 731 :
0262 732 :
0262 733 :      discon  connect  mailbox
0262 734 :      reason status  message
0262 735 :      -----
0262 736 :
0262 737 MRC  NORMAL, REJECT, REJECT ; Connect reject
0268 738 MRC  EXIT, LINKEXIT, EXIT ; User exit or timeout
026E 739 MRC  NOPATH, UNREACHABLE, PATHLOST ; Path lost to partner node
0274 740 MRC  SHUT, SHUT, NETSHUT ; Node shutting down
027A 741 MRC  NOBJ, NOSUCHOBJ, REJECT ; No such object
0280 742 MRC  ABORT, LINKABORT, ABORT ; Disconnect abort
0286 743 MRC  THIRD, THIRDPARTY, THIRDPARTY ; Disconnect by third party
028C 744 MRC  ACCESS, INVLOGIN, REJECT ; Login info invalid
0292 745 MRC  RSU, REMRSRC, REJECT ; Resource error
0298 746 MRC  BUSY, REMRSRC, REJECT ; Object too busy
029E 747 MRC  FMT, PROTOCOL, REJECT ; Illegal process name field
02A4 748 MRC  NONODE, NOSUCHNODE, REJECT ; Unrecognized node i.d.
02AA 749 MRC  IVNODE, NOSUCHNODE, REJECT ; Invalid node-i.d. format
02B0 750 :
02B0 751 :
02B0 752 :      The following are internal codes and are not part of NSP
02B0 753 :
02B0 754 :
02B0 755 MRC  DEACC, ABORT, ABORT ; Link is IO$ DEACCESS'ed
02B6 756 MRC  INVALID, CONNCFail, ABORT ; Reason field never setup
02BC 757 :
FFFFF 02BC 758 .LONG -1 ; Terminate the table (the last entry
02C0 759 ; is to be used as a catch-all)
02C0 760 :
02C0 761 :
02C0 762 :
02C0 763 NET$MAP_R REASON:: ; Map Reason code in XWBSW_R REASON
50 FF3A CF 9E 02C0 764 -MOVAB W^NET$AZ_DR_TABLE - ; Setup non-connect table address
02C5 765 -REASON_C_LENGTH,RO ; ...biased for scan
02C5 766 :
02C5 767 ASSUME XWBS_C_STA_CLO EQ 0
02C5 768 ASSUME XWBS_C_STA_CIS EQ 1
02C5 769 ASSUME XWBS_C_STA_CAR EQ 2
02C5 770 :
1E A5 02 91 02C5 771 CMPB #2,XWBSB_STA(R5) ; Is 'connect initiate' table needed?
05 19 02C9 772 BLSS 10$ ; If LSS then no
50 FF8D CF 9E 02CB 773 MOVAB W^NET$AZ_DR_CONTAB - ; Setup connect-initiate table address
02D0 774 -REASON_C_LENGTH,RO ; ...biased for scan
02D0 775 :
50 06 C0 02D0 776 10$: ADDL #REASON_C_LENGTH,RO ; Goto next entry
44 A5 B1 02D3 777 CMPW XWBSW_R_REASON(R5),- ; Does it match?
60 02D6 778 REASON_Q_DR(RO)
07 13 02D7 779 BEQL 20$ ; If EQL then yes
60 D5 02D9 780 TSTL (RO) ; At end of table?
F3 18 02DB 781 BGEQ 10$ ; If GEQU then no
50 06 C2 02DD 782 SUBL #REASON_C_LENGTH,RO ; No match found, use the default entry
05 02E0 783 20$: RSB
02E1 784
```

NETDRVSES
V04-000

F 13
- DECnet Session Control Module for NETD 16-SEP-1984 01:32:10 VAX/VMS Macro V04-00
NET\$AZ_DR_TABLE - Disconnect Reason Code 5-SEP-1984 02:20:26 [NETACP.SRC]NETDRVSES.MAR;1 Page 19
(27)

02E1 785
02E1 786

NE
VC

NETDRVSES
V04-000

G 13

- DECnet Session Control Module for NETD 16-SEP-1984 01:32:10 VAX/VMS Macro V04-00
NET\$AZ_CR_TABLE - Disconnect Reason Code 5-SEP-1984 02:20:26 [NETACP.SRC]NETDRVSES.MAR;1 Page 20
(29)

	02E1	788	
	02E1	789	NET\$INTERRUPT:
	02E1	790	NET\$CTLR INIT:
05	02E1	791	RSB
	02E2	792	NET\$UNIT INIT:
01	02E2	793	NOP
01	02E3	794	NOP
01	02E4	795	NOP
01	02E5	796	NOP
01	02E6	797	NOP
01	02E7	798	NOP
01	02E8	799	NOP
01	02E9	800	NOP
01	02EA	801	NOP
05	02EB	802	RSB
	02EC	803	

```
02EC 805 .SBTTL NET$FORK - Fork the XWB to do new work
02EC 806 :+
02EC 807 :
02EC 808 : If the fork block in the XWB is available, it is forked so that the work
02EC 809 : in XWB$W_FLG will be done. If the fork block is unavailable, no further
02EC 810 : action is required since the XWB$W_FLG work will get done when the XWB fork
02EC 811 : process is subsequently resumed.
02EC 812 :
02EC 813 :
02EC 814 : INPUTS: R5 XWB address
02EC 815 : R0 Garbage
02EC 816 :
02EC 817 : OUTPUTS: R0 #1
02EC 818 :
02EC 819 : All other registers are preserved
02EC 820 :
02EC 821 :-
02EC 822 NET$FORK::
06 0E A5 02 E2 02EC 823 BBSS #XWB$V_STS_SOL,XWB$W_STS(R5),20$ : Fork the XWB
02F1 824 : : If BS, fork block in use
02F1 825 : :
02F3 826 : : Save regs
02F5 827 10$: POPR #^M<R1,R2,R3,R4,R5> : Schedule fork and return
02F7 828 : :
02F7 829 20$: MOVL #1,R0 : Always return success
02FA 830 : : Done
02FB 831 : :
02FB 832 30$: MOVAB XWB$Q_FORK(R5),R5 : Switch to fork block context
02FF 833 : : Fork
0305 834 : : Restore XWB context
0309 835 : :
0309 836 : : We're back
030D 837 : : Clear wait flag to allow retry
0311 838 : : If BC, disconnect not pending
0316 839 : :
0316 840 : : Save Event regs
031A 841 : : Say 'not okay to go to IPL 2'
031C 842 : : Event is 'resume deaccess'
031F 843 : : Signal the event
0322 844 : : Restore regs
0326 845 : :
0326 846 100$: BSBW NET$SCH_MSG : Schedule message transmission
0329 847 : : Done
032A 848 : :
032A 849 : :
```

06 0E A5 02 E2 02EC 823 BBSS #XWB\$V_STS_SOL,XWB\$W_STS(R5),20\$: Fork the XWB
02F1 824 : : If BS, fork block in use
02F1 825 : : Save regs
02F3 826 : : Schedule fork and return
02F5 827 10\$: POPR #^M<R1,R2,R3,R4,R5> :
02F7 828 : :
02F7 829 20\$: MOVL #1,R0 : Always return success
02FA 830 : : Done
02FB 831 : :
02FB 832 30\$: MOVAB XWB\$Q_FORK(R5),R5 : Switch to fork block context
02FF 833 : : Fork
0305 834 : : Restore XWB context
0309 835 : :
0309 836 : : We're back
030D 837 : : Clear wait flag to allow retry
0311 838 : : If BC, disconnect not pending
0316 839 : :
0316 840 : : Save Event regs
031A 841 : : Say 'not okay to go to IPL 2'
031C 842 : : Event is 'resume deaccess'
031F 843 : : Signal the event
0322 844 : : Restore regs
0326 845 : :
0326 846 100\$: BSBW NET\$SCH_MSG : Schedule message transmission
0329 847 : : Done
032A 848 : :
032A 849 : :

```
032A 851 .SBTTL NET$END_EVENT - Abort current event without changing state
032A 852 .SBTTL NET$COMPLEX_EV - Change state and process new event
032A 853 .SBTTL NET$PRE_EMPT - Process new event without changing state
032A 854 :+
032A 855 :
032A 856 : These routines are called by the dispatched event action routines in order
032A 857 : to complete current event processing in a non-standard way. They should be
032A 858 : considered substitutes to the RSB instruction which is normally used to
032A 859 : return control -- consequently the stack is checked for the return address
032A 860 : of the event dispatcher.
032A 861 :
032A 862 : CALLING SEQUENCE: JMP NET$xxx
032A 863 :
032A 864 : INPUTS: R10 Preserved for call to action routine
032A 865 : R9 The value originally stored by the event dispatcher
032A 866 : R8 Preserved for call to action routine
032A 867 : R7 Code of new event to be processed (scratch if NET$END_EVENT)
032A 868 : R6 The value originally stored by the event dispatcher
032A 869 : R5 XWB address
032A 870 : R4-R1 Scratch
032A 871 : R0 If NET$END_EVENT the status to be returned to the
032A 872 : caller of the event dispatcher,
032A 873 : Else scratch
032A 874 : (SP) The address of CHANGE_STA which is the NET$EVENT return
032A 875 : address.
032A 876 :
032A 877 : OUTPUTS: N/A
032A 878 :
032A 879 :-
032A 880 NET$END_EVENT::
032A 881 BSBB CHKRETADDR ; End event without changing state
032A 882 BSBW NET$SCH_MSG ; Make sure stack is setup properly
032A 883 RSB ; Schedule message transmission
032A 884
032A 885 NET$COMPLEX_EV::
032A 886 BSBB CHKRETADDR ; Change state, process new event
032A 887 EXTZV #NET$C_ACTBITS,- ; Validate state of stack
032A 888 #NET$C_STABITS,R9,R4 ;
032A 889 CMPB R4,XWB$B_STA(R5) ; Get next state
032A 890 BEQL 10$ ; New state ?
032A 891 BSBB NEW STATE ; If not, branch
032A 892 10$: BRB NET$EVENT ; Enter new state
032A 893 ; Process new event
032A 894 NET$PRE_EMPT::
032A 895 BSBB CHKRETADDR ; Pre-empt the current event
032A 896 BRB NET$EVENT ; Validate state of stack
032A 897 ; Process new event
032A 898 CHKRETADDR:
032A 899 PUSHAB B^CHANGE_STA ; Checks return address to trap bugs
032A 900 CMPL (SP)+,4(SP) ; Prepare for next instruction
032A 901 BNEQ 5$ ; Is state of stack correct?
032A 902 MOVL (SP)+,(SP) ; If NEQ then no
032A 903 RSB ; Overlay return address
032A 904 ; Return
032A 905 5$: BUG_CHECK NETNOSTATE,FATAL
032A 906
```

19	10	032A	881	BSBB	CHKRETADDR	; End event without changing state			
0074	30	032C	882	BSBW	NET\$SCH_MSG	; Make sure stack is setup properly			
	05	032F	883	RSB		; Schedule message transmission			
		0330	884						
		0330	885	NET\$COMPLEX_EV::		; Change state, process new event			
	13	10	0330	886	BSBB	CHKRETADDR	; Validate state of stack		
	05	EF	0332	887	EXTZV	#NET\$C_ACTBITS,-			
54	59	03	0334	888		#NET\$C_STABITS,R9,R4	; Get next state		
1E	A5	54	0337	889	CMPB	R4,XWB\$B_STA(R5)	; New state ?		
		02	033B	890	BEQL	10\$	; If not, branch		
		3F	033D	891	BSBB	NEW STATE	; Enter new state		
		15	033F	892	10\$: BRB	NET\$EVENT	; Process new event		
			0341	893					
		02	0341	894	NET\$PRE_EMPT::		; Pre-empt the current event		
		11	0341	895	BSBB	CHKRETADDR	; Validate state of stack		
			0343	896	BRB	NET\$EVENT	; Process new event		
			0345	897					
			0345	898	CHKRETADDR:		; Checks return address to trap bugs		
04	AE	6F 'AF	9F	0345	899	PUSHAB	B^CHANGE_STA	; Prepare for next instruction	
		8E	D1	0348	900	CMPL	(SP)+,4(SP)	; Is state of stack correct?	
		04	12	034C	901	BNEQ	5\$	; If NEQ then no	
		6E	8E	D0	034E	902	MOVL	(SP)+,(SP)	; Overlay return address
			05	0351	903	RSB		; Return	
				0352	904				
				0352	905	5\$: BUG_CHECK	NETNOSTATE,FATAL		
				0356	906				

```
0356 908 .SBTTL NET$EVENT - Event dispatcher
0356 909 :+
0356 910 :
0356 911 : This is the state table event dispatcher used to determine what is to be
0356 912 : done and what state the XWB is to enter next. An event only has meaning
0356 913 : within the context of a XWB.
0356 914 :
0356 915 :
0356 916 : INPUTS: R10 Preserved for call to action routine
0356 917 : R9 Available for the event dispatcher's exclusive use
0356 918 : R8 Preserved for call to action routine
0356 919 : R7 Code of the event to be processed
0356 920 : R6 If received message event then Transport's IRP address
0356 921 : If "startio" event then UCB address
0356 922 : R5 Address of XWB
0356 923 : R4 Scratch
0356 924 : R3 If received message event then scratch
0356 925 : If "startio" event then QIO IRP address
0356 926 : R2 If received msg event then message bytes not yet accounted f
0356 927 : If "startio" event then scratch
0356 928 : R1 If received msg event then ptr to first unprocessed byte in
0356 929 : If "startio" event then scratch
0356 930 : R0 Scratch
0356 931 :
0356 932 : OUTPUTS: R0 Status code from the action routine to be returned to
0356 933 : the caller of the event dispatcher.
0356 934 :
0356 935 : Only R6 and R5 are preserved.
0356 936 :
0356 937 :-
0356 938 :
0356 939 ASSUME XWB$C_NUMSTA EQ 8 ; Assume quadword per event
0356 940 NET$EVENT:: ; Process new event
0356 941 MOVZBL XWB$B_STA(R5),R9 ; Get current state
0356 942 MOVAQ NET$AB_STTAB[R7],R4 ; Get event block address
0356 943 MOVZBL (R4)[R9],R9 ; Get table entry
0356 944 BICL3 #NET$M_STAMSK,R9,R4 ; Get action routine index
0356 945 :
0356 946 :
0356 947 : Dispatch according to the event code. The action routines
0356 948 : can assume the following :
0356 949 :
0356 950 : Inputs:
0356 951 :
0356 952 : R10 Parameter from caller to action routine
0356 953 : R9 State information -- not to be touched
0356 954 : R8 Parameter from caller to action routine
0356 955 : R7 Event code
0356 956 : R6 Varies with event
0356 957 : R5 XWB address
0356 958 : R4 Scratch
0356 959 : R3-R1 Varies with event
0356 960 : R0 Scratch
0356 961 : (SP) Return address
0356 962 :
0356 963 : Returned values:
0356 964 :
```

```
54 59 000000E0 8F 9A 0356 941
54 59 000000E0 8F 7E 035A 942
54 59 000000E0 8F 9A 0360 943
54 59 000000E0 8F CB 0364 944
```

```
036C 965 : R8,R7 Garbage
036C 966 : R6,R5 Preserved
036C 967 : R4-R1 Garbage
036C 968 : R0 Status to be returned to caller of dispatcher
036C 969 :
036C 970 :
00D8 30 036C 971 BSBW ACT_DISPATCH ; Call action routine
036F 972
036F 973 CHANGE_STA: ; Change logical-link state
54 59 05 EF 036F 974 EXTZV #NET$C_ACTBITS,- ;
1E A5 03 0371 975 #NET$C_STABITS,R9,R4 ; Get next state
54 91 0374 976 CMPB R4,XWBSB_STA(R5) ; New state ?
29 13 0378 977 BEQL NET$SCH_MSG ; If EQL no, schedule message xmission
02 10 037A 978 BSBB NEW_STATE ; Change to new state
25 11 037C 979 BRB NET$SCH_MSG ; Schedule message xmission
037E 980
037E 981
037E 982 NEW_STATE: ; Change to new logical-link state
037E 983
037E 984
037E 985 Clear PROGRESS since we are changing states. The only exceptions
037E 986 are if we are coming out of the "closed" state (since PROGRESS has
037E 987 been setup to the correct value by the previous action routine) or
037E 988 if we are entering the "connect ACK received" state (since we the
037E 989 outgoing link timeout period is tied to the receipt of a Connect
037E 990 Confirm, not a Connect-ACK).
037E 991
037E 992
00 91 037E 993 CMPB #XWBS_C_STA_CLO,- ; Coming out of the "closed" state?
1E A5 0380 994 XWBSB_STA(R5)
08 13 0382 995 BEQL 10$ ; If EQL, PROGRESS was already init'd
54 02 91 0384 996 CMPB #XWBS_C_STA_CAR,R4 ; Entering "CAR" state?
03 13 0387 997 BEQL 10$ ; If EQL yes, do not re-init PROGRESS
52 A5 B4 0389 998 CLRW XWBSW_PROGRESS(R5) ; Init progress count
1E A5 54 90 038C 999 10$: MOVB R4,XWBSB_STA(R5) ; Change state
AA 0390 1000 BICW #XWBSM_STS_TID,- ; Always clear "timer,id valid" and
0391 1001 XWBSM_STS_DIS,- ; "disconnect pending" flags
0391 1002 XWBSW_STS(R5)
0394 1003
1C A5 FD93 CF44 AA 0394 1004 BICW NET$AW_FLG_CLRM[R4],XWBSW_FLG(R5) ; Clear indicated flags
1C A5 FD7C CF44 AB 039B 1005 BISW NET$AW_FLG_SETM[R4],XWBSW_FLG(R5) ; Set indicated flags
05 03A2 1006 RSB ; Done
03A3 1007
```

```
03A3 1009 .SBTTL NET$SCH_MSG - schedule message transmission
03A3 1010 :+
03A3 1011 :
03A3 1012 : The following flags are used to cause control messages to be setup when the
03A3 1013 : control message cell in the XWB becomes available. As each message is
03A3 1014 : entered into this control message cell, the corresponding bit is cleared.
03A3 1015 :
03A3 1016 : These flags are listed in the order of their priority.
03A3 1017 :
03A3 1018 :
03A3 1019 : XWBSV_FLG_TBPR - Set whenever the receive back pressure state needs to
03A3 1020 : be toggled.
03A3 1021 :
03A3 1022 : XWBSV_FLG_IAVL - Set whenever a new xmit interrupt IRP makes it to the
03A3 1023 : head of the LSB queue and the partner's flow control
03A3 1024 : on the INT/LS subchannel will let us send the message.
03A3 1025 :
03A3 1026 : XWBSV_FLG_SIFL - Set whenever an INTERRUPT message has been sent to the
03A3 1027 : user's mailbox.
03A3 1028 :
03A3 1029 : XWBSV_FLG_SDFL - Set whenever the inactivity timer fires in order to
03A3 1030 : maintain a minimal amount of traffic on the link to
03A3 1031 : see if the remote node is still active.
03A3 1032 :
03A3 1033 :
03A3 1034 : Whether or not a new Link-service/Interrupt message is setup in the XWB
03A3 1035 : cell, XWBSW_FLG(R5) is scanned to see if any work needs to be done. If
03A3 1036 : so, and if the XWB fork block is not in use, control is passed to
03A3 1037 : NET$SOLICIT.
03A3 1038 :
03A3 1039 :
03A3 1040 : INPUTS: R5 XWB address
03A3 1041 : R4-R0 Scratch
03A3 1042 :
03A3 1043 : OUTPUTS: R4-R0 Garbage
03A3 1044 :
03A3 1045 : All other registers are preserved.
03A3 1046 :
03A3 1047 :-
03A3 1C 3 NET$SCH_MSG:: ; Schedule message transmission
03A3 1049 :
03A3 1050 : ASSUME XWBSV_FLG_IAVL EQ 1+XWBSV_FLG_TBPR
03A3 1051 : ASSUME XWBSV_FLG_SIFL EQ 1+XWBSV_FLG_IAVL
03A3 1052 : ASSUME XWBSV_FLG_SDFL EQ 1+XWBSV_FLG_SIFL
03A3 1053 :
03A3 1054 : FFS #XWBSV_FLG_TBPR,#4,XWBSW_FLG(R5),R0 ; Find message to build
03A3 1055 : BEQL 90$ ; If EQL then none
03A3 1056 : BBS #NSP$V_FLW_INUSE,XWBSB_X_FLW(R5),90$ ; If BS, msg cell is in use
03A3 1057 : MOVB #NSP$M_FLW_INUSE,XWBSB_X_FLW(R5) ; Claim the cell, clear flags
03A3 1058 : CLRB XWBSB_X_FLWCNT(R5) ; Init flow request count
03A3 1059 :
03A3 1060 : MOVAB XWBSW_LI(R5),R2 ; Setup LSB pointer
03A3 1061 : INCW LSB$W_LUX(R2) ; Get next sequence number
03A3 1062 : BICW #^X<F000>,LSB$W_LUX(R2) ; Mask off junk bits
03A3 1063 : MOVW LSB$W_LUX(R2),[SB$W_HXS(R2) ; It's sendable now
03A3 1064 : BBCC R0,XWBSW_FLG(R5),20$ ; Clear the work bit
03A3 1065 :
```

50	1C	A5	04	0B	EA
				74	13
	6F	6C	A5	04	E0
		6C	A5	10	90
			6D	A5	94
	52	00D4	C5	9E	03B7
			62	B6	03BC
	62	F000	8F	AA	03BE
		04	A2	62	B0
	00	1C	A5	50	E5
				03C7	1064
				03CC	1065

				03CC	1066	20\$:	\$DISPATCH R0,-		; Dispatch on work bit
				03CC	1067		<-		
				03CC	1068		<XWBSV_FLG_I AVL, 50\$>,-		; INTerrupt msg
				03CC	1069		<XWBSV_FLG_SIFL, 40\$>,-		; INTerrupt flow control msg
				03CC	1070		<XWBSV_FLG_SDFL, 80\$>,-		; DATA flow control msg
				03CC	1071		>		; Fall thru if
				03D6	1072				; R0=XWBSV_FLG_TBPR
				03D6	1073				
				03D6	1074				
				03D6	1075				
				03D6	1076				
				03D6	1077				
				03D6	1078				
				03D6	1079		MOVB #NSPSM_FLW_XOFF,R0		; Setup for "stop flow" message
13 OE	50	01	90	03D9	1080		BBCS #XWBSV_STS_RBP,XWBSW_STS(R5),30\$		; If BC, not back-pressured off
OE A5	A5	06	E3	03DE	1081		BICW #XWBSM_STS_RBP,XWBSW_STS(R5)		; Mark receiver as having its
			AA	03E4	1082				; back-pressured relaxed
	50	02	90	03E4	1083		MOVB #NSPSM_FLW_XON,R0		; Setup for "start flow" msg
				03E7	1084				
				03E7	1085				
				03E7	1086				
				03E7	1087				
				03E7	1088				
				03E7	1089				
				03E7	1090				
				03E7	1091				
				03E7	1092				
				03E7	1093				
				03E7	1094				
				03E7	1095				
				03E7	1096				
				03E7	1097				
				03E7	1098				
				03E7	1099				
				03E7	1100				
				03E7	1101				
				03E7	1102				
				03E7	1103				
				03E7	1104		BISW #XWBSM_FLG_SDACK,XWBSW_FLG(R5)		; Force NAK on the DATA channel
				03EB	1105		BISW #XWBSM_STS_DTNAK,XWBSW_STS(R5)		; in order to reset it
OE A5	1C A5	08	A8	03F1	1106	30\$:	BISB R0,XWBSB_X_FLW(R5)		; Set remaining control flags
	6C A5	50	88	03F5	1107		BBCC #XWBSV_FLG_SIFL,XWBSW_FLG(R5),80\$		; Piggy-back INT flow control
	1B 1C	A5	OD	03FA	1108				; message if possible
				03FA	1109	40\$:			
				03FA	1110				
				03FA	1111				
				03FA	1112				
				03FA	1113				
				03FA	1114		BISB #NSPSM_FLW_LISUB,XWBSB_X_FLW(R5)		; Flow control for LI channel
6C A5	6D A5	04	88	03FE	1115		INCB XWBSB_X_FLCNT(R5)		; Ask for one more INT message
	00FD	C5	96	0401	1116		INCB XWBSB_X_FLCNT(R5)		; And allow it to be received
		OE	11	0405	1117	50\$:	BRB 80\$		; Schedule msg for transmission
				0407	1118				
				0407	1119				
				0407	1120				
				0407	1121				
				0407	1122				

```
      6C A5 20 88 0407 1123      BISB #NSP$M_FLW_INT,XWB$B_X_FLW(R5) ; Not 'link service' message
      50 10 A2 D0 040B 1124      MOVL LSB$X_PND(R2),R0 ; Get IRP
20 A0 0040 8F AA 040F 1125      BICW #IOSM_INTERRUPT,IRP$W_FUNC(R0) ; Indicate state change
      0415 1126 80$:          :
      0415 1127          :
      0415 1128          : Schedule the message for transmission.
      0415 1129          :
      0415 1130          :
      1C A5 10 A8 0415 1131      BISW #XWB$M_FLG_SLI,XWB$W_FLG(R5) ; We've got a message to send
      4000 8F AA 0419 1132      BICW #XWB$M_FLG_SDFL,XWB$W_FLG(R5) ; Whatever has just been built
      041F 1133          : satisfies the need to send the
      041F 1134          : background inactivity message
      041F 1135 90$:          :
      041F 1136          :
      041F 1137          : Unless there are any wait conditions pending, solicit permission
      041F 1138          : from the Routing layer to transmit a message.
      041F 1139          :
      041F 1140          :
      0A 00 EA 041F 1141      FFS #0,#XWB$V_FLG_CLO+1,- ; Get work bit
      50 1C A5 0422 1142      XWB$W_FLG(R5),R0 ;
OE FD BA CF 50 E1 0425 1143      BBC R0,NET$GC_WORKBITS,200$ ; Br if no work to be done
      08 0E A5 02 E2 042B 1144      BBSS #XWB$V_STS_SOL,XWB$W_STS(R5),100$ ; If BS, fork block in use
      55 DD 0430 1145          :
      FBCB' 30 0432 1147      PUSHL R5 ; Save XWB address
      55 8E D0 0435 1148      BSBW NSP$SOLICIT ; Get permission to transmit
      05 0438 1149 100$:      POPL R5 ; Restore XWB address
      0439 1150          :
      FA 0E A5 03 E1 0439 1151 200$:      BBC #XWB$V_STS_DIS,XWB$W_STS(R5),100$ ; If BC, disconnect not pending
      0381 30 043E 1152      BSBW NET$CHK_X_IDLE ; Is XWB ready for disconnect ?
      F4 50 E9 0441 1153      BLBC R0,100$ ; If LBC then no
      FEA5 31 0444 1154      BRW NET$FORK ; Attempt to resume disconnect
      0447 1155          :
```

```
0447 1157
0447 1158 ACT_DISPATCH: ; Dispatch action routine
0447 1159
0447 1160 $DISPATCH TYPE=B,R4, -; R4 contains the action index
0447 1161 <-
0447 1162 <ACT$_ABORT, ACT$ABORT>, -;
0447 1163 <ACT$_BUG, ACT$BUG>, -;
0447 1164 <ACT$_CANLNK, ACT$CANLNK>, -;
0447 1165 <ACT$_CONFIRM, ACT$CONFIRM>, -;
0447 1166 <ACT$_DEACCESS, ACT$DEACCESS>, -;
0447 1167 <ACT$_ENT_RUN, ACT$ENT_RUN>, -;
0447 1168 <ACT$_INITIATE, ACT$INITIATE>, -;
0447 1169 <ACT$_LOG, ACT$LOG>, -;
0447 1170 <ACT$_NOP, ACT$NOP>, -;
0447 1171 <ACT$_RES_DISC, ACT$RES_DISC>, -;
0447 1172 <ACT$_RCV_CA, ACT$RCV_CA>, -;
0447 1173 <ACT$_RCV_CC, ACT$RCV_CC>, -;
0447 1174 <ACT$_RCV_CI, ACT$RCV_CI>, -;
0447 1175 <ACT$_RCV_CR, ACT$RCV_CR>, -;
0447 1176 <ACT$_RCV_DATA, ACT$RCV_DATA>, -;
0447 1177 <ACT$_RCV_DTACK, ACT$RCV_DTACK>, -;
0447 1178 <ACT$_RCV_DX, ACT$RCV_DX>, -;
0447 1179 <ACT$_RCV_LI, ACT$RCV_LI>, -;
0447 1180 <ACT$_RCV_LIACK, ACT$RCV_LIACK>, -;
0447 1181 <ACT$_RCV_RTS, ACT$RCV_RTS>, -;
0447 1182 <ACT$_RTS_NLT, ACT$RTS_NLT>, -;
0447 1183 <ACT$_SHR[ENK, ACT$SHR[ENK>, -;
0447 1184 <ACT$_SSABORT, ACT$SSABORT>, -;
01 11 0447 1185 >
0477 1186 BRB ACT$BUG ; If unknown, bug
0479 1187
```

```
0479 1189 .SBTTL ACT$NOP          - Null action routine
0479 1190 .SBTTL ACT$BUG          - BUG_CHECK action routine
0479 1191 .SBTTL ACT$LOG          - Log-event action routine
0479 1192 .SBTTL ACT$NOLINK       - Report 'SS$ FILNOTACC'
0479 1193 .SBTTL ACT$SSABORT      - Abort QIO since link was disconnected
0479 1194
05 0479 1195 ACT$NOP:            RSB
047A 1196 ACT$BUG:              BUG_CHECK NETNOSTATE,FATAL
01 047E 1197 ACT$LOG:            NOP
01 047F 1198                    NOP
0480 1199
01 0480 1200                    NOP
01 0481 1201                    NOP
01 0482 1202                    NOP
05 0483 1203                    RSB
0484 1204
0484 1205
38 A3 00AC 8F 3C 0484 1206 ACT$NOLINK: MOVZWL #SS$_FILNOTACC,IRP$L_IOST1(R3)
05 048A 1207 RSB
0489 1208
048B 1209 ACT$SHRLNK:           ;&nyi
38 A3 2C 3C 048B 1210 ACT$SSABORT: MOVZWL #SS$_ABORT,IRP$L_IOST1(R3)
05 048F 1211 RSB
049C 1212
0490 1213
```

```
0490 1215 .SBTTL NET$STARTIO - Start I/O operation
0490 1216 :+
0490 1217 :
0490 1218 : This routine is entered when the associated unit is idle and a packet
0490 1219 : is available for processing. The IRP$L_WIND field is used to locate the
0490 1220 : associated XWB.
0490 1221 :
0490 1222 :
0490 1223 : INPUTS: R5 UCB address
0490 1224 : R4 PCB address
0490 1225 : R3 IRP address
0490 1226 :
0490 1227 : OUTPUTS: *** TBS ***
0490 1228 :
0490 1229 :
0490 1230 NET$STARTIO:
0490 1231 PUSHHR #M<R5,R6,R7,R8,R9,R10,R11> ; Process next IRP
0490 1232 BSBB PROC_IO ; Setup 'event' context
0490 1233 BLBS R5,20$ ; Process the I/O function
0490 1234 CLRL R11 ; If LBS, no event to process
0490 1235 BSBW NET$EVENT ; Say 'can't go to IPL 2'
0490 1236 20$: POPR #M<R5,R6,R7,R8,R9,R10,R11> ; Process event in R7
0490 1237 ; Return to UCB 'fork' context
0490 1238 MOVL UCB$L_IRP(R5),R3 ; Get IRP
0490 1239 BEQL 50$ ; If EQL then its been queued
0490 1240 ; or suspended, start next I/O
0490 1241 :
0490 1242 :
0490 1243 : Deallocate misc. buffer
0490 1244 :
0490 1245 :
0490 1246 BBC #IRP$V_COMPLX,IRP$W_STS(R3),40$ ; If BC, IRP$L_DIAGBUF does not
0490 1247 ; point to a NETDRIVER buffer
0490 1248 MOVL IRP$L_DIAGBUF(R3),R0 ; Get buffer
0490 1249 BGEQ 40$ ; If GEQ then none
0490 1250 CLRL IRP$L_DIAGBUF(R3) ; Detach it
0490 1251 BSBW NET$DEALLOCATE ; Deallocate block in R0
0490 1252 40$:
0490 1253 :
0490 1254 : Start next I/O.
0490 1255 :
0490 1256 :
0490 1257 MOVL IRP$L_IOST1(R3),R0 ; First IOSB longword
0490 1258 MOVL UCB$L_DEVDEPEND(R5),R1 ; Second IOSB longword
0490 1259 REQCOM ; Complete I/O, start next IRP
0490 1260 50$: REMQUE @UCB$L_IOQFL(R5),R3 ; Get next IRP
0490 1261 BVS 60$ ; If VS then none
0490 1262 JMP G^IOCS$INITIATE ; Deliver IRP to driver
0490 1263 60$: BICW #UCB$M_BSY,UCB$W_STS(R5) ; Free up the UCB
0490 1264 RSB ; Return to Exec
0490 1265
0490 1266 PROC_IO:
0490 1267 :
0490 1268 : Move the UCB to R6 and the XWB (if any) to R5 and dispatch
0490 1269 : on function code with:
0490 1270 :
0490 1271 : R10-R7 Scratch
```

```

      04DA 1272      :      R6      UCB address
      04DA 1273      :      R5      XWB address if LBC, else garbage
      04DA 1274      :      R3      IRP address
      04DA 1275      :      R2-R0   Scratch
      04DA 1276      :
      04DA 1277      :
      56 55 D0 04DA 1278      MOVL    R5,R6      ; Copy UCB to safe register
      55 18 A3 D0 04DD 1279      MOVL    IRP$L_WIND(R3),R5      ; Get XWB, if any
      03 19 04E1 1280      BLSS    10$      ; If LSS, XWB is in system space
      55 01 C8 04E3 1281      BISL    #1,R5      ; Else, invalidate window ptr
      38 A3 00F4 8F 3C 04E6 1282 10$:      MOVZWL #$$$_ILLIOFUNC,IRP$L_IOST1(R3) ; Assume fct not supported
      57 20 A3 FFFFFFFC 8F CB 04EC 1283      BICL3 #^C<IOSM_FCODE>,IRP$Q_FUNC(R3),R7 ; Get code without modifiers
      04F5 1284
      04F5 1285      $DISPATCH R7,TYPE=B,-      ; Process I/O
      04F5 1286      <-
      04F5 1287      <IOS_ACCESS, NET$ACCESS>,-      ; Connect Requests
      04F5 1288      <IOS_DEACCESS, NET$DEACCESS>,-      ; Disconnect Requests
      04F5 1289      <IOS_ACPCONTROL, NET$CONTROL>,-      ; ACP Control function
      55 01 D0 0507 1291      >      ; Else, fall thru
      05 050A 1292      MOVL    #1,R5      ; Set low bit to prevent event
      050B 1293      RSB      ; dispatching and return
```

```
050B 1295 .SBTTL NET$FDT_SETMODE - Process IO$_SETMODE request
050B 1296 :+
050B 1297 :
050B 1298 : *** tbs ***
050B 1299 :-
050B 1300 :
050B 1301 NET$FDT_SETMODE: ; Process IO$_SETMODE function
51 6C D0 050B 1302 MOVL P1(AP),R1 ; Get characteristics buffer
OB 13 050E 1303 BEQL 10$ ; If EQL then none
44 A5 04 A1 D0 0510 1304 IFNORD #8,(R1),50$ ; Br on access violation
50 18 A3 01 CB 0516 1305 MOVL 4(R1),UCB$L DEVDEPEND(R5) ; Set mailbox mask
00 18 051B 1306 10$: BICL3 #1,IRP$L_WIND(R3),R0 ; Get XWB address
0520 1307 BGEQ 40$ ; If GEQ then none
0522 1308 :
0522 1309 :
0522 1310 : This was used for 'maintenance' mode. Now available for
0522 1311 : future functions.
0522 1312 :
0522 1313 :
51 44 A5 D0 0522 1314 40$: MOVL UCB$L DEVDEPEND(R5),R1 ; Get device dependent info
50 01 3C 0526 1315 MOVZWL S^#SS$ NORMAL,R0 ; Setup I/O status
00000000'GF 17 0529 1316 JMP G^EXE$FINISHIO ; Return success
50 0C 3C 052F 1317 50$: MOVZWL #SS$ ACCVIO,R0 ; Setup I/O status
00000000'GF 17 0532 1319 JMP G^EXE$ABORTIO ; Abort I/O
0538 1320
```

```
0538 1322 .SBTTL NET$FDT_CONTROL - IOS_ACPCONTROL FDT processing
0538 1323 .SBTTL NET$CONTROL - IOS_ACPCONTROL "startio" processing
0538 1324
0538 1325
0538 1326 The FDT routine simply routes the IRP through the Exec to the ACP. The Exec
0538 1327 builds a "complex buffer" describing the control function. The ACP will
0538 1328 requeue any IRP to the driver if it does not recognize the control function.
0538 1329 The driver has been designed to handle some of its own control functions
0538 1330 since many are protocol or control block format specific.
0538 1331
0538 1332
0538 1333 INPUTS: R5 UCB Address
0538 1334 R4 PCB Address
0538 1335 R3 IRP Address
0538 1336
0538 1337 OUTPUTS: R5 Unchanged
0538 1338 R0 I/O status
0538 1339
0538 1340
0538 1341 NET$FDT_CONTROL:
0538 1342 CLRL IRP$L_DIAGBUF(R3) ; FDT phase for IOS_ACPCONTROL
0538 1343 BICL #1,IRP$L_WIND(R3) ; Zero misc buffer pointer
0538 1344 ASSUME PHD$Q_PRIVMSK EQ 0 ; Always clear interlock flag
0538 1345 MOVQ @PCB$[PHD(R4),IRP$Q_NT_PRIVMSK(R3) ; Store privilege mask
0538 1346 JMP G^ACPS$MODIFY ; Continue in EXEC
0538 1347
0538 1348
0538 1349
0538 1350 INPUTS: *** tbs ***
0538 1351 OUTPUTS: *** tbs ***
0538 1352
0538 1353
0538 1354 NET$CONTROL:
0538 1355 BBC #IRP$V_COMPLX,IRP$W_STS(R3),10$ ; "Startio" for IOS_ACPCONTROL
0538 1356 MOVL @IRP$L_SVAPTE(R3),R0 ; If BC, part of $CANCEL
0538 1357 CLRL (R0) ; Get ptr to window descriptor
0538 1358 BRB 50$ ; Don't affect window upon
0538 1359 10$: ; I/O completion
0538 1360
0538 1361
0538 1362 The user is getting ready to issue an IOS_DEACCESS QIO to break the
0538 1363 link. In order for the IOS_DEACCESS to be sent to the driver, the
0538 1364 channel's outstanding I/O count (CCBSW_IOC) must be zero. Hence the
0538 1365 receiver must be run-down and any outstanding receive IRP's aborted.
0538 1366
0538 1367
0538 1368 MOVZWL #SS$ NORMAL,IRP$L_IOST1(R3) ; Set I/O status
0538 1369 BICL #1,R5 ; Clear interlock bit
0538 1370 BLSS 70$ ; If SS then valid XWB
0538 1371
0538 1372
0538 1373 Scan LTB to find XWB with an access pending for this channel
0538 1374
0538 1375
0538 1376 MOVL UCB$L_VCB(R6),R0 ; Get RCB
0538 1377 BEQL 50$ ; If EQL then none
0538 1378 MOVL RCB$L_PTR_LTB(R0),R0 ; Get LTB
0538 1379 BEQL 50$ ; If EQL then none
```

18 A3 4C A3 D4 01 CA
40 A3 6C B4 7D 00000000'GF 17
08 2A A3 03 E1
50 2C B3 D0
60 D4
1F 11
38 A3 01 3C
55 01 CA
2F 19
50 34 A6 D0
10 13
50 24 A0 D0
0A 13

```

55 E0 A0 9E 056C 1379 MOVAB -XWBSL_LINK+LTBSL_XWB(R0),R5 ; Prepare for XWB scan
55 2C A5 D0 0570 1380 20$: MOVL XWBSL_LINK(R5),R5 ; Get next XWB
      04 12 0574 1381 BNEQ 60$ ; If EQL then none left
      55 01 88 0576 1382 50$: BISB #1,R5 ; Prevent event dispatching
      05 0579 1383 RSB ; Done
      057A 1384
50 0080 C5 D0 057A 1385 60$: MOVL XWBSL_IRP_ACC(R5),R0 ; Get suspended IRP, if any
      EF 13 057F 1386 BEQL 20$ ; If EQL none, loop
OC A3 0C A0 D1 0581 1387 CMPL IRP$L_PID(R0),IRP$L_PID(R3) ; Belong to this process ?
      E8 12 0586 1388 BNEQ 20$ ; Loop if NEQ
28 A3 28 A0 B1 0588 1389 CMPW IRP$W_CHAN(R0),IRP$W_CHAN(R3) ; Same channel ?
      E1 12 058D 1390 BNEQ 20$ ; If NEQ, loop
      058F 1391 70$:
      058F 1392
      058F 1393
      058F 1394
      058F 1395
      058F 1396
      058F 1397
      058F 1398
      058F 1399
      058F 1400
      058F 1401
      058F 1402
      058F 1403
      058F 1404
      058F 1405
      058F 1406
      058F 1407
      058F 1408
      058F 1409
      058F 1410
      058F 1411
      058F 1412 80$:
      05A2 1413
      05A3 1414
      05A3 1415

The transmitter is not automatically run-down since the user may be
preparing a "synchronous" disconnect -- i.e., disconnect after the
final data segment has been ACK'd. The manner in which pipelining
has been implemented allows user transmit IRP's to be sent to I/O
completion before the corresponding CXB's have been ACK'd (or even
sent). Therefore, the user might issue a call to $CANCEL mistakenly
thinking that the final message has been ACK'd. Hence $CANCEL should
allow the transmit CXB's to be ACK'd in their normal fashion.

Therefore, drain the receiver of all IRP's and CXB's. If there
are any transmit IRP's on the queue, then the disconnect is not
synchronous, and the transmitter queue must be drained as well.

OB OE A5 04 E0 058F 1407 BBS #XWBSV_STS_CON,XWBSW_STS(R5),80$ ; If BS, IOS_ACCESS pending
      038A 30 0594 1408 BSBW DRAIN_RCV ; Drain the Receiver
      50 D4 0597 1409 CLRL R0 ; Init R0 for CHK_X_IRP call
      0235 30 0599 1410 BSBW CHK_X_IRP ; Any Xmt IRP's
      D7 50 E8 059C 1411 BLBS R0,50$ ; If LBS, no
      57 0D 3C 059F 1412 MOVZWL #NETEVT$_CANLNK,R7 ; Force link to break
      05 05A2 1413 RSB ; Done
      05A3 1414
      05A3 1415
```

```
05A3 1417 .SBTTL NET$FDT_ACCESS - IOS_ACCESS FDT processing
05A3 1418 .SBTTL NET$ACCESS - IOS_ACCESS 'startio' processing
05A3 1419 ++
05A3 1420 :
05A3 1421 : NET$FDT_ACCESS passes the IRP through the EXEC, where the user parameters
05A3 1422 : are packaged into a "complex buffer", to the ACP. The ACP reads the user
05A3 1423 : connect info to build an Internal Connect Block (ICB) which it attaches to
05A3 1424 : the IRP$DIAGBUF field of the IRP and queues the IRP to the driver. The
05A3 1425 : role of the ACP is to lookup default access control (username, password,
05A3 1426 : account) information in its data base and to translate node and object names
05A3 1427 : to numbers.
05A3 1428 :
05A3 1429 : NET$ACCESS reads the ICB and determines the type of connect. It builds an
05A3 1430 : XWB for connect initiate events and locates an already existing XWB for all
05A3 1431 : others. NET$ACCESS stores the appropriate event code in R7 and returns
05A3 1432 : expecting the caller to call the event dispatcher.
05A3 1433 :
05A3 1434 : It should be noted that the size of the XWB is not charged against the user
05A3 1435 : byte count or byte limit quotas. It is assumed that these quotas are at
05A3 1436 : least partly used to limit a run away process and that the file quota of a
05A3 1437 : process, against which logical-links are charged, is a sufficient mechanism.
05A3 1438 :
05A3 1439 :
05A3 1440 :
05A3 1441 : INPUTS: *** tbs ***
05A3 1442 :
05A3 1443 : OUTPUT: *** tbs ***
05A3 1444 :
05A3 1445 :
05A3 1446 :
05A3 1447 NET$FDT_ACCESS: ; IOS_ACCESS 'FDT' processing
05A3 1448 :
05A3 1449 : ASSUME PHD$Q_PRIVMSK EQ 0
05A3 1450 :
40 A3 6C B4 7D 05A3 1451 MOVQ @PCBSL_PHD(R4),IRP$Q_NT_PRIVMSK(R3) ; Store priv mask in IRP
4C A3 D4 05A8 1452 CLRL IRP$DIAGBUF(R3) ; Indicate no ICB
00000000'GF 17 05AB 1453 JMP G*ACP$ACCESSNET ; Continue in EXEC
05B1 1454 :
05B1 1455 :
05B1 1456 NET$ACCESS: ; IOS_ACCESS 'startio' processing
05B1 1457 BSBW GET_WNDSC ; Get CCBSL_WIND image descr.
05B4 1458 CLRL (R7) ; Init CCBSL_WIND image
05B6 1459 BBSW #IRP$M_FUNC,IRP$W_STS(R3) ; Mark for write back
32 A3 02 A8 05BA 1460 MOVZWL #1,IRP$DIAGBUF(R3) ; Write back one descriptor
55 56 D0 05BE 1461 MOVL R6,R5 ; Copy UCB addr for subroutines
58 53 D0 05C1 1462 MOVL R3,R8 ; Copy IRP address to save reg
54 4C A8 D0 05C4 1463 MOVL IRP$DIAGBUF(R8),R4 ; Get ICB pointer
3C 18 05C8 1464 BGEQ 80$ ; If GEQ, its an error code
05CA 1465 :
05CA 1466 :
05CA 1467 : IOS_ACCESS made it successfully through the ACP
05CA 1468 :
05CA 1469 :
05CA 1470 MOVZWL ICB$W_LOCLNK(R4),R3 ; Get local link address
53 02 A4 3C 05CE 1471 BSBW XWB_LOCLNK ; Find associated XWB
06C6 30 05D1 1472 BLBS R5,80$ ; Br if XWB was not found
2D 55 E8 05D4 1473 CMPL XWB$DIAGBUF(R5),IRP$DIAGBUF(R8) ; PIDs match ?
OC A8 34 A5 D1
```

```

      1F 12 05D9 1474      BNEQ 55$      ; Br if they don't
3C A5 B5 05DB 1475      TSTW XWB$W_REMLNK(R5) ; Does remote link id exit ?
      OB 12 05DE 1476      BNEQ 30$      ; Connect Confirm if NEQ
      05E0 1477
      05E0 1478
      05E0 1479
      05E0 1480
      05E0 1481
      05E0 1482      ADDW3 #1,ICB$W_TIM_OCON(R4),- ; Setup outbound connect timer
04 A4 01 A1 05E4 1483      XWB$W_TIMER(R5) ; (+1 for possible clock skew)
      50 A5 05E6 1484      MOVZBL #NETEVT$_CIA,R7 ; Set 'connect initiate access'
      57 OF 9A 05E9 1485      BRB 50$      ; Finish in common
      OB 11 05EB 1486 30$:
      05EB 1487
      05EB 1488
      05EB 1489
      05EB 1490
      05EB 1491 40$:      MOVZBL #NETEVT$_CCA,R7 ; Set 'connect confirm access'
03 20 A8 08 E1 05EE 1492      BBC #IOSV_ABORT,IRP$W_FUNC(R8),50$ ; If BC, not Connect Reject
      57 11 9A 05F3 1493      MOVZBL #NETEVT$_CRA,R7 ; Set 'connect reject access'
      05F6 1494 50$:
      05F6 1495
      05F6 1496
      05F6 1497
      05F6 1498
      05F6 1499
      53 58 D0 05F6 1500      MOVL R8,R3 ; Setup IRP address
      05 05F9 1501      RSB ; Return with LBC in R5
      05FA 1502
      05FA 1503
      05FA 1504
      05FA 1505
      05FA 1506 55$:      MOVZWL #SS$_DEVALLOC,R4 ; Setup error code
      05FF 1507      BRB 80$      ; Continue
      0601 1508 60$:      MOVZWL #SS$_CONNECFAIL,R4 ; Setup error code
      D0 0606 1509 80$:      MOVL R8,R3 ; Setup IRP pointer
      D0 0609 1510      MOVL R4,IRP$L_IOST1(R3) ; Store error code
      D0 060D 1511      MOVL #1,R5 ; Tell CLEANUP_ACCESS 'no XWB'
      38 A3 54 D0 0610 1512      BSBW CLEANUP_ACCESS ; Restore quota
      55 01 30 0613 1513      RSB ; On return goto REQCOM
      0201 05 0614 1514
      0614 1515 100$:      BUG_CHECK NETNOSTATE,FATAL
      0618 1516
```

```
0618 1518 .SBTTL ACT$INITIATE - Connect Initiate action routine
0618 1519 .SBTTL ACT$CONFRM - Connect Confirm action routine
0618 1520 +
0618 1521
0618 1522 These action routines resume processing the event setup by NET$ACCESS.
0618 1523 ACT$INITIATE assumes that a Connect Initiate message will be built
0618 1524 and sent. ACT$CONFRM is used when a received connect is being either
0618 1525 accepted or rejected and assumes that either a Connect Confirm or a
0618 1526 Disconnect Initiate message will be built and sent.
0618 1527
0618 1528
0618 1529 INPUTS: R8 Scratch
0618 1530 R7 Event code
0618 1531 R6 UCB address
0618 1532 R5 XWB address
0618 1533 R4 Scratch
0618 1534 R3 IRP address
0618 1535 R2-R0 Scratch
0618 1536
0618 1537 OUTPUTS: R8,R7 Garbage
0618 1538 R6,R5 Preserved
0618 1539 R4-R0 Garbage
0618 1540
0618 1541 -
0618 1542 ACT$CONFRM:: ; Connect Confirm or Reject
35 10 0618 1543 BSBB SETUP_XWB ; Do common setup
061A 1544
061A 1545
061A 1546 If the remote end of the Logical-link is on the local node then
061A 1547 use the same 'path'. This allows loopbacked lines to be used for
061A 1548 all logical-link traffic in both directions -- which seems like a
061A 1549 sensible thing to do even though this may be a departure from the
061A 1550 Network Management architecture.
061A 1551
061A 1552
061A 1553 TSTW XWBSW_PATH(R5) ; Has a path been chosen ?
061D 1554 BNEQ 20$ ; If NEQ then yes
OE 52 30 A5 D0 061F 1555 MOVL XWBSL_VCB(R5),R2 ; Get the RCB
A2 3A A5 B1 0623 1556 CMPW XWBSW_REMNOD(R5),RCBSW_ADDR(R2) ; Is the remote node us?
16 12 0628 1557 BNEQ 20$ ; If NEQ no
062A 1558
062A 1559 PUSHR #^M<R3,R4,R5> ; Save regs
53 3C A5 3C 062C 1560 MOVZWL XWBSW_REMLNK(R5),R3 ; Get remote link i.d.
0676 30 0630 1561 BSBW NET$XWB_LOCLNK ; Find corresponding XWB
52 55 D0 0633 1562 MOVL R5,R2 ; Copy XWB address to new reg
38 BA 0636 1563 POPR #^M<R3,R4,R5> ; Restore regs
0638 1564
0638 1565 BLBS R2,20$ ; If LBS then no XWB was found
38 A5 05 52 E8 0638 1566 MOVW XWBSW_PATH(R2),XWBSW_PATH(R5) ; Use partner's path i.d.
09 20 A3 08 B0 063B 1567 BBC #IOSV_ABORT,IRPSW_FUNC(R3),100$ ; If BC then not connect reject
0645 1568 ASSUME NET$C_DR NORMAL EQ 0
46 A5 B4 0645 1569 CLRW XWBSW_X_REASON(R5) ; Setup disconnect reason code
50 01 7D 0648 1570 MOVQ S^#SS$ NORMAL,R0 ; Setup IOSB value
007C 30 064B 1571 BSBW NET$CPL_ACC ; Complete the IOS_ACCESS IRP
05 064E 1572 100$: RSB ; Done
064F 1573
064F 1574 ACT$INITIATE:: ; Connect Initiate request
```

```
016C C5 0C A5 B6 064F 1575 SETUP_XWB: ; Setup common fields
      34 A5 D0 064F 1576 INCW XWBSW_REFCNT(R5) ; Account for accessor
0080 C5 53 D0 0652 1577 MOVL XWBSL_PID(R5),XWBSW+ACBSL_PID(R5) ; Setup Special Kernal AST PID
      58 A6 D4 0658 1578 MOVL R3,XWBSL_IRP_ACC(R5) ; Store IRP address
      10 A5 56 D4 065D 1579 CLRL UCBSL_IRP(R6) ; Clear IRP pointer to prevent
      54 4C A3 D0 0660 1580 ; immediate I/O completion
      4C A3 D0 0660 1581 MOVL R6,XWBSL_ORGUCB(R5) ; Setup UCB ptr
      010C C5 54 D0 0664 1582 MOVL IRP$DIAGBUF(R3),R4 ; Get ICB ptr
1C A5 0100 8F A8 0668 1583 CLRL IRP$DIAGBUF(R3) ; Detach it from IRP
      0670 1584 MOVL R4,XWBSL_ICB(R5) ; Attach it to XWB
      0676 1585 BISW #XWBSM_FCG_SCD,XWBSW_FLG(R5) ; Set send message flag
      0676 1586 30$: ;
      0676 1587 ;
      0676 1588 ; Setup pre-allocated byte quota to take upon entering the RUN state
      0676 1589 ;
      0676 1590 ;
      0676 1591 :8 CLRW XWBSW_X_QUO(R5) ; Pre-allocate none for rcv's
      0676 1592 :8 CLRW XWBSW_R_QUO(R5) ; Pre-allocate none for rcv's
      0676 1593 ;
      0676 1594 ;
      0676 1595 ; Move remainder of parameters from the ICB
      0676 1596 ;
      0676 1597 ;
      38 BB 0676 1598 PUSHR #^M<R3,R4,R5> ; Save MOVC regs
      51 7C A4 9E 0678 1599 ;
      06E0 30 0678 1600 MOVAB ICB$B_DATA(R4),R1 ; Get source pointer
      067F 1601 BSBW NET$MOV_TO_XWB ; Move counted string
      067F 1602 ; to XWBSB_DATA...
      6F A5 0092 C4 90 067F 1603 ASSUME ICB$C_RID LE XWBS$C_RID
      20 0093 C4 10 2C 067F 1604 MOVAB ICB$B_RID(R4),XWBSB_RID(R5) ; Move the count field
      70 A5 10 0685 1605 MOVAB #ICB$C_RID,ICB$T_RID(R4),#^A' ', - ; Move the remote
      068B 1607 #XWBS$C_RID,XWBS$T_RID(R5) ; i.d. text
      068E 1608 ;
      50 38 BA 068E 1609 POPR #^M<R3,R4,R5> ; Restore regs
      0C A4 B0 0690 1610 MOVW ICB$W_RETRAN(R4),R0 ; Get rexmt factor
      04 15 0694 1611 BLEQ 50$ ; If LEQ keep default
      54 A5 50 B0 0696 1612 MOVW R0, XWBSW_RETRAN(R5) ; Update rexmt factor
      52 A5 B4 069A 1613 50$: ; Init progress count
      38 A5 64 B0 069D 1614 MOVW ICB$W_PATH(R4), XWBSW_PATH(R5) ; Circuit to use
      40 A5 12 A4 B0 06A1 1615 MOVW ICB$W_SEGSIZ(R4), XWBSW_LOCSIZ(R5) ; Rcv buffer size
      56 A5 0E A4 B0 06A6 1616 MOVW ICB$W_DLY_FACT(R4), XWBSW_DLY_FACT(R5) ; Delay factor
      58 A5 10 A4 B0 06AB 1617 MOVW ICB$W_DLY_WGHT(R4), XWBSW_DLY_WGHT(R5) ; Delay weight
      4C A5 06 A4 B0 06B0 1618 MOVW ICB$W_TIM_INACT(R4),XWBSW_TIM_INACT(R5) ; Inactivity timer
      06B5 1619 ;
      06B5 1620 ;
      06B5 1621 ; Setup TIMER and DELAY so that the Connect message will be
      06B5 1622 ; retransmitted periodically if necessary. This is done by choosing
      06B5 1623 ; a DELAY which will allow RETRAN retransmission before the amount
      06B5 1624 ; of time left in TIMER expires.
      06B5 1625 ;
      06B5 1626 ;
      54 A5 A7 06B5 1627 DIVW3 XWBSW_RETRAN(R5),- ; TIMER has number of ticks
      50 A5 06B8 1628 XWBSW_TIMER(R5),XWBSW_DELAY(R5) ; left before timeout
      56 A5 A6 06BC 1629 DIVW XWBSW_DLY_FACT(R5),- ; Adjust for the "delay factor"
      4E A5 06BF 1630 XWBSW_DELAY(R5) ;
      03 12 06C1 1631 BNEQ 70$ ; If NEQ then use it
```

NETDRVSES
V04-000

M 14
- DECnet Session Control Module for NETD 16-SEP-1984 01:32:10 VAX/VMS Macro V04-00 Page 39
ACT\$CONFRIM - Connect Confirm action rou 5-SEP-1984 02:20:26 [NETACP.SRC]NETDRVSES.MAR;1 (41)

4E A5	B6	06C3	1632	INCW	XWBSW_DELAY(R5)	; Else use 1 second
F937	30	06C6	1633	BSBW	NET\$RESET_TIMER	; Reset XWBSW_TIMER
	05	06C9	1634	RSB		; Done
		06CA	1635			

```
06CA 1637 .SBTTL NET$CMPL_ACC - Complete IO$_ACCESS, fill in window
06CA 1638 :++
06CA 1639 :
06CA 1640 : The access function currently being processed is completed.
06CA 1641 : If the I/O completion status is not successful then the window of the
06CA 1642 : channel associated with the IRP is cleared.
06CA 1643 :
06CA 1644 :
06CA 1645 : INPUTS: R5 XWB address
06CA 1646 : R1 Second IOSB longword value
06CA 1647 : R0 First IOSB longword value
06CA 1648 :
06CA 1649 : OUTPUTS: R1 Garbage
06CA 1650 : R0 $$$_NORMAL
06CA 1651 :
06CA 1652 : All other registers are preserved.
06CA 1653 :
06CA 1654 :
06CA 1655 NET$CMPL_ACC::
06CA 1656 PUSH R2,R3,R4,R5,R7,R8 : Complete access, fill in window
06CE 1657 : Save regs
06CE 1658 :
06CE 1659 CMPW XWB$_DELAY(R5),#4 : Make sure initial 'delay' estimate
06D2 1660 BGEQU 30$ : is at least 4 seconds
06D4 1661 MOVW #4,XWB$_DELAY(R5)
06D8 1662 MOVL XWB$_IRP_ACC(R5),R3 : Get IO$_ACCESS IRP
06DD 1663 BEQL 200$ : If EQL then none
06DF 1664 CLRL XWB$_IRP_ACC(R5) : Remove IRP
06E3 1665 MOVQ R0,IRP$_IOST1(R3) : Save I/O status
06E7 1666 :
06E7 1667 : Setup the CCB$_WIND value and deallocate the ICB.
06E7 1668 :
06E7 1669 : If either the access was unsuccessful or the request was for a
06E7 1670 : Connect Reject, then cleanup from the IO$_ACCESS attempt and
06E7 1671 : leave the CCB$_WIND image at zero.
06E7 1672 :
06E7 1673 :
06E7 1674 BBS #IO$_V_ABORT,- : If BS, Connect Reject
06E9 1675 IRP$_FUNC(R3),60$ :
06EC 1676 CLRB XWB$_DATA(R5) : Init optional data cell to prepare
06EF 1677 : for eventual disconnect
06EF 1678 BLBS R0,100$ : If LBS then successful IO$_ACCESS
06F2 1679 BSBW CLEANUP_ACCESS : Cleanup from access I/O fct
06F5 1680 BRB 110$ : Complete the I/O
06F7 1681 BSBW GET_WNDSC : Get CCB$_WIND image descriptor
06FA 1682 MOVL R5,(R7) : Setup CCB$_WIND value
06FD 1683 BSBW DEAL_ICB : Deallocate the ICB
0700 1684 110$:
0700 1685 :
0700 1686 : Complete the I/O
0700 1687 :
0700 1688 :
0700 1689 BSBW NET$POST_IO : Post IRP for completion
0703 1690 :
0703 1691 POPR #M<R2,R3,R4,R5,R7,R8> : Restore regs
0707 1692 MOVL $$$_NORMAL,R0 : Success
070A 1693 RSB
```

01BC 8F BB
04 4E A5 B1
4E A5 04 1E
53 0080 C5 B0
0080 C5 D0
38 A3 50 D4
06 20 08 E0
5B A5 94
05 50 E8
011F 30
09 11
00F4 30
67 55 D0
0161 30
067D 30
01BC 8F BA
50 01 D0
05 070A

NETDRVSES
V04-000

B 15
- DECnet Session Control Module for NETD 16-SEP-1984 01:32:10 VAX/VMS Macro V04-00
NET\$CMPL_ACC - Complete IO\$_ACCESS, fill 5-SEP-1984 02:20:26 [NETACP.SRC]NETDRVSES.MAR;1 Page 41
0708 1694
0708 1695 (42)

N
V

```

070B 1697 .SBTTL ACT$ENT_RUN - Enter RUN state action routine
070B 1698 :+
070B 1699 :
070B 1700 : This routine is entered to setup the XWB for entering the RUN state.
070B 1701 :
070B 1702 : INPUTS: R7 Event code - it will be reprocessed via the complex event
070B 1703 : mechanism. Note that the state should have
070B 1704 : been updated by then.
070B 1705 : R5 XWB address
070B 1706 : R0 Scratch
070B 1707 :
070B 1708 : OUTPUTS: R0 garbage
070B 1709 :
070B 1710 : All other registers are preserved.
070B 1711 :
070B 1712 :-
070B 1713 ACT$ENT_RUN::
F8F2' 30 070B 1714 BSBW NET$SETUP_RUN ; Enter RUN state
FC1F 31 070E 1715 BRW NET$COMPLEX_EV ; Setup XWB for RUN state
0711 1716 ; Change state and resignal the event

```

```
0711 1718 .SBTTL NET$FDT_DEACCESS- IOS_DEACCESS FDT processing
0711 1719 .SBTTL NET$DEACCESS - IOS_DEACCESS 'startio' processing
0711 1720 :++
0711 1721 :
0711 1722 : INPUTS: AP Pointer to the QIO P1 parameter
0711 1723 : R8 Must be saved/restored if return to Exec for next
0711 1724 : FDT routine
0711 1725 : R7 I/O function code without modifiers
0711 1726 : R6 CCB address
0711 1727 : R5 UCB address
0711 1728 : R4 PCB address
0711 1729 : R3 IRP address
0711 1730 : R2-R0 Scratch
0711 1731 :
0711 1732 : OUTPUTS: R5,R3 Preserved
0711 1733 :
0711 1734 : All other regs may be clobbered.
0711 1735 :
0711 1736 :--
0711 1737 NET$FDT_DEACCESS::
0711 1738 : IOS_DEACCESS FDT routine
0711 1739 : Zero misc buffer pointer
0711 1740 : Assume 'link not accessed'
0711 1741 : Clear interlock bit
0711 1742 : If GEQ, link is not accessed
0711 1743 :
0711 1744 : Save regs
0711 1745 : Synchronize
0711 1746 : Say 'okay to go to IPL 2'
0711 1747 : Switch to XWB context
0711 1748 :
0711 1749 :
0711 1750 : Setup disconnect reasons codes as appropriate
0711 1751 :
0711 1752 :
0711 1753 : CMPW #NETSC_DR_INVALID,XWBSW_R_REASON(R5) ; Rcv'd reason code yet ?
0711 1754 : BNEQ 10$ ; If NEQ yes
0711 1755 : MOVZWL #NETSC_DR_DEACC, XWBSW_R_REASON(R5) ; Setup local reason
0711 1756 : CMPW #NETSC_DR_INVALID,XWBSW_X_REASON(R5) ; Xmt reason code setup ?
0711 1757 : BGTRU 20$ ; If GTRU, yes
0711 1758 : MOVZWL #NETSC_DR_NORMAL, XWBSW_X_REASON(R5) ; Assume ordinary disconn.
0711 1759 : BBC #IOSV_ABORT, IRPSW_FUNC(R3),20$ ; If BS, must abort all I/O
0711 1760 : MOVZWL #NETSC_DR_ABORT, XWBSW_X_REASON(R5) ; Else, set 'disc. abort'
0711 1761 : 20$:
0711 1762 :
0711 1763 : If IOSV_ABORT is set, then both the transmitter and receiver must
0711 1764 : be run-down. Otherwise, this is a 'synchronous' disconnect and
0711 1765 : the transmitter must be allowed to send all data before breaking
0711 1766 : the link.
0711 1767 :
0711 1768 :
0711 1769 : BBS #XWBSV_STS_CON, XWBSW_STS(R5),100$ ; If BS, not in RUN format
0711 1770 : BBC #IOSV_ABORT, IRPSW_FUNC(R3),50$ ; If BS, must abort all I/O
0711 1771 : BSBW DRAIN_XMT ; Run-down the transmitter
0711 1772 : BSBW DRAIN_RCV ; Run-down the receiver
0711 1773 :
0711 1774 : 100$: ENBINT ; Restore IPL
```

50 4C A3 D4 0711 1738
00AC 8F 3C 0714 1739
18 A3 01 CA 0719 1740
52 18 071D 1741
0F78 8F BB 071F 1742
5B 01 D0 0723 1744
072A 1745
55 18 A3 D0 072D 1746
072D 1747
0731 1748
0731 1749
0731 1750
0731 1751
0731 1752
44 A5 0064 8F B1 0731 1753
06 12 0737 1754
44 A5 0066 8F 3C 0739 1755
46 A5 0064 8F B1 073F 1756 10\$:
0D 1A 0745 1757
46 A5 00 3C 0747 1758
04 20 A3 08 E1 074B 1759
46 A5 09 3C 0750 1760
0754 1761 20\$:
0754 1762
0754 1763
0754 1764
0754 1765
0754 1766
0754 1767
0754 1768
0B 0E A5 04 E0 0754 1769
03 20 A3 08 E1 0759 1770
0161 30 075E 1771
01BD 30 0761 1772 50\$:
0764 1773
0764 1774 100\$: ENBINT

```
0F78 8F BA 0767 1775 POPR #*M<R3,R4,R5,R6,R8,R9,R10,R11> ; Restore regs
                                076B 1776 ;
00000000'GF 17 076B 1777 JMP G*ACPSDEACCESS ; Goto system FDT routine
00000000'GF 17 0771 1778 200$: JMP G*EXESABORTIO ; Abort the I/O
                                0777 1779 ;
                                0777 1780 ;
                                0777 1781 NET$DEACCESS:: ; User QIO to break link
0074 30 0777 1782 BSBW GET WNDSC ; Get CCB$L_WIND image desc
67 D4 077A 1783 CLRL (R7) ; Clear CCB$L_WIND image
38 A3 01 3C 077C 1784 MOVZWL #SS$ NORMAL,IRP$L_IOST1(R3) ; Setup success status
2A A3 02 A8 0780 1785 BISW #IRP$M_FUNC,IRP$W_STS(R3) ; Mark for write back
32 A3 01 D0 0784 1786 MOVL #1,IRP$L_BCNT(R3) ; Write back 1 (the window) ABD
57 12 D0 0788 1787 MOVL #NETEVT$_DEA,R7 ; Setup event code in case R5
05 078B 1788 RSB ; is a valid XWB pointer
078C 1789 ;
078C 1790 ;
078C 1791 ACT$DEACCESS:: ; User QIO to break link
0085 30 078C 1792 BSBW CLEANUP_ACCESS ; Clean up from access I/O fct
0065 30 078F 1793 BSBW GET_P2DSC ; Get optional data descriptor
58 D7 0792 1794 DECL R8 ; Reduce length by count field
1C 19 0794 1795 BLSS ACT$RES_DISC ; If LSS, then no data
51 67 9A 0796 1796 MOVZBL (R7),R1 ; Get count value from string
51 58 D1 0799 1797 CMPL R8,R1 ; Take minimum of size from
03 1F 079C 1798 BLSSU 20$ ; descriptor and size from
58 51 D0 079E 1799 MOVL R1,R8 ; string
10 58 D1 07A1 1800 20$: CMPL R8,#16 ; Take minimum of what's there
03 1F 07A4 1801 BLSSU 30$ ; and max allowed by NSP
58 10 D0 07A6 1802 MOVL #16,R8 ;
67 58 90 07A9 1803 30$: MOVB R8,(R7) ; Setup count field in string
51 57 D0 07AC 1804 MOVL R7,R1 ; Setup source ptr
05AD 30 07AF 1805 BSBW NET$MOV_TO_XWB ; Move counted string to
07B2 1806 ; XWB$B_DATA
07B2 1807 ACT$RES_DISC:: ; Resume Disconnect processing
07B2 1808 ;
07B2 1809 ;
07B2 1810 ; If the XWB is idle, continue processing this event. Else, dismiss
07B2 1811 ; this event for now and resume it when the XWB goes idle. This is
07B2 1812 ; the only way to do a "synchronous disconnect" with NSP pipelining
07B2 1813 ; causing user Transmit IRP's to be completed before the CXB's are
07B2 1814 ; actually transmitted.
07B2 1815 ;
07B2 1816 ;
0E 10 07B2 1817 BSBB NET$CHK_X_IDLE ; Is the transmitter idle ?
04 50 E9 07B4 1818 BLBC R0,100$ ; If LBS then no
00F5 30 07B7 1819 BSBW NET$PURG_RUN ; Cleanup if necessary
05 07BA 1820 RSB ; Return to change state
07BB 1821 ;
0E A5 08 A8 07BB 1822 100$: BISW #XWB$M_STS_DIS,XWB$W_STS(R5) ; Mark disconnect pending
FB68 31 07BF 1823 BRW NET$END_EVENT ; Dismiss this event for now
07C2 1824 ;
```

```
07C2 1826
07C2 1827 .ENABL LSB
07C2 1828
07C2 1829 NET$CHK_X_IDLE:: ; See if transmitter is idle
20 OE A5 04 E0 07C2 1830 BBS #XWB$V_STS_CON,XWB$W_STS(R5),10$ ; If BS, not in RUN format
50 00BC C5 D0 07C7 1831 MOVL XWB$T_DT+LSB$X_CXB(R5),R0 ; Get next DATA CXB
50 00EC C5 C8 07CC 1832 BISL XWB$T_LI+LSB$X_CXB(R5),R0 ; OR in next Interrupt CXB
50 00B4 C5 C8 07D1 1833 CHK_X_IRP: ; Check for Xmt IRP's
50 00B8 C5 C8 07D6 1834 BISL XWB$T_DT+LSB$X_PND(R5),R0 ; OR in pending DATA Xmt IRP's
50 00E4 C5 C8 07D8 1835 BISL XWB$T_DT+LSB$X_IRP(R5),R0 ; OR in spent DATA Xmt IPR's
50 00B8 C5 C8 07DB 1836 BISL XWB$T_LI+LSB$X_PND(R5),R0 ; OR in pending Int. Xmt IRP's
50 00B8 C5 C8 07E0 1837 BISL XWB$T_DT+LSB$X_IRP(R5),R0 ; OR in spent Int. Xmt IPR's
50 04 12 07E5 1838 BNEQ 20$ ; If NEQ then not idle
50 01 D0 07E7 1839 10$: MOVL #1,R0 ; Say 'idle'
05 07EA 1840 RSB ; Done
50 07EB 1841
50 D4 07EB 1842 20$: CLRL R0 ; Say 'non-idle'
05 07ED 1843 RSB ; Done
07EE 1844
07EE 1845 .DSABL LSB
07EE 1846
07EE 1847
07EE 1848 .ENABL LSB
58 D4 07EE 1849 GET_WNDSC: ; Get window descriptor
12 11 07EE 1850 CLRL R8 ; Get descriptor offset
07F0 1851 BRB 10$ ; Continue
58 08 D0 07F2 1852 GET_P1DSC: ; Get P1 descriptor
0D 11 07F2 1853 MOVL #8,R8 ; Get descriptor offset
07F5 1854 BRB 10$ ; Continue
58 10 D0 07F7 1855 GET_P2DSC: ; Get P2 descriptor
08 11 07F7 1856 MOVL #8*2,R8 ; Get descriptor offset
07FA 1857 BRB 10$ ; Continue in common
58 18 D0 07FC 1858 GET_P3DSC: ; Get P3 descriptor
03 11 07FC 1859 MOVL #8*3,R8 ; Get descriptor offset
07FF 1860 BRB 10$ ; Continue in common
58 20 D0 0801 1861 GET_P4DSC: ; Get P4 descriptor
58 2C B3 C0 0801 1862 MOVL #8*4,R8 ; Get descriptor offset
57 88 3C 0804 1863 10$: ADDL @IRP$L_SVAPTE(R3),R8 ; Get descriptor address
57 FF A847 9E 0808 1864 MOVZWL (R8)+,R7 ; Get offset to data
080B 1865 MOVAB -1(R8)[R7],R7 ; Get ptr to data after skipping
58 68 3C 0810 1866 MOVZWL (R8),R8 ; over access mode byte
05 0810 1867 RSB ; Get length of data
0813 1868
0814 1869 .DSABL LSB
0814 1870
```

```
0814 1872 .SBTTL CLEANUP_ACCESS - Cleanup XWB for terminated IOS_ACCESS
0814 1873 :
0814 1874 :
0814 1875 : INPUTS:      R5 - XWB address, low bit set if none
0814 1876 :            R3 - IRP address
0814 1877 :
0814 1878 : OUTPUTS:     All registers are preserved.
0814 1879 :
0814 1880 :
0814 1881 :
0814 1882 CLEANUP_ACCESS:
0814 1883   PUSHHR      #*M<R0,R1,R2>                ; Save regs
0816 1884   :
0816 1885   CLRL      R2                            ; Assume no byte quota to return
19 55 0818 1886   BLBS      R5,20$                ; If LBS then no XWB
04 44 0818 1887   BSBB      DEAL_ICB                ; Deallocate the ICB
04 F6 081D 1888   BSBW      NET$DRAIN FREE CXB        ; Drain CXB free queue
0C A5 0820 1889   DECW      XWB$W_REFCT(R5)            ; Account for loss of accessor
08 38 0823 1890   BNEQ      200$                    ; Br if last accessor
10 A5 0825 1891 10$:   CLRL      XWB$$_ORGUCB(R5)        ; XWB is unowned
34 A5 0828 1892   CLRL      XWB$$_PID(R5)              ; XWB is unowned
04 0E A5 0A E0 0828 1893   BBS      #XWB$V_STS_ASTPND,XWB$W_STS(R5),20$ ; If BS, AST block in use
01 6C C5 D4 0830 1894   CLRL      XWB$$_ACB$$_PID(R5)    ; Prevent false AST delivery
0834 1895 20$:   :
0834 1896   :
0834 1897   : Return BYTCNT and FILCNT quota
0834 1898   :
0834 1899   :
51 50 0C A3 3C 0834 1900   MOVZWL  IRP$$_PID(R3),R0        ; Get PID index
00000000'GF D0 0838 1901   MOVL     G$SCH$GL_PCBVEC,R1    ; Get PCB vector address
51 61 40 D0 083F 1902   MOVL     (R1)[R0],R1              ; Get PCB address
0C A3 60 A1 D1 0843 1903   CMPL     PCB$$_PID(R1),IRP$$_PID(R3) ; Still there?
10 12 0848 1904   BNEQ      30$                        ; If not branch
50 00 80 C1 D0 084A 1905   MOVL     PCB$$_JIB(R1),R0        ; Get JIB
30 A0 B6 084F 1906   INCW      JIB$$_FILCNT(R0)          ; Return quota for IOS_ACCESS
20 A0 52 C0 0852 1907   ADDL      R2,JIB$$_BYTCNT(R0)      ; Return byte quota
24 A0 52 C0 0856 1908   ADDL      R2,JIB$$_BYTLM(R0)       ; Here too
085A 1909   :
07 BA 085A 1910 30$:   POPR      #*M<R0,R1,R2>          ; Restore regs
05 085C 1911   RSB                ; Done
085D 1912   :
085D 1913 200$:   BUG_CHECK NETNOSTATE,FATAL            ; Invalid reference count
0861 1914   :
0861 1915 DEAL_ICB:   : Deallocate the ICB
07E 50 7D 0861 1916   MOVQ      R0,-(SP)                ; Save regs
0864 1917   :
0E 0E A5 04 E1 0864 1918   BBC      #XWB$V_STS_CON,XWB$W_STS(R5),40$ ; If BC, XWB$$_ICB is invalid
50 01 0C C5 D0 0869 1919   MOVL     XWB$$_ICB(R5),R0      ; Get buffer for deallocation
03 18 086E 1920   BGEQ      30$                        ; If GEQ then none
04 DB 30 0870 1921   BSBW      NET$DEALLOCATE            ; Deallocate block in R0
01 0C C5 D4 0873 1922 30$:   CLRL      XWB$$_ICB(R5)        ; Say 'no ICB'
0877 1923   :
50 8E 7D 0877 1924 40$:   MOVQ      (SP)+,R0              ; Restore regs
05 087A 1925   RSB                ; Done
087B 1926   :
```

```
087B 1928 .SBTTL NET$CANCEL - Cancel I/O routine
087B 1929 :
087B 1930 :
087B 1931 : Most of the work for the Cancel-I/O sequence will occur when the special
087B 1932 : IOS_ACPCONTROL QIO is issued by the $CANCEL system service.
087B 1933 :
087B 1934 : In all cases, the ACP is informed via a mailbox message since special
087B 1935 : cleanup may be needed in the ACP (e.g. declared name cleanup). Note that
087B 1936 : the special Cancel IRP is only sent to the ACP if there is a logical-link
087B 1937 : active.
087B 1938 :
087B 1939 :
087B 1940 : INPUTS: R5 UCB address
087B 1941 : R4 PCB address
087B 1942 : R3 IRP address if UCB is busy
087B 1943 : R2 Channel number
087B 1944 : R1,R0 Scratch
087B 1945 :
087B 1946 : NET$IPL
087B 1947 :
087B 1948 : OUTPUTS: R3-R0 Garbage
087B 1949 :
087B 1950 : All other registers are preserved
087B 1951 :
087B 1952 :
087B 1953 : NET$CANCEL:
50 34 A5 D0 087B 1954 : MOVL UCB$L_VCB(R5),R0 ; Cancel I/O entry point
24 13 087F 1955 : BEQL 50$ ; Get VCB address
0881 1956 : ; If EQL then none
0881 1957 :
0881 1958 : Tell the ACP
0881 1959 :
0881 1960 :
01BC 8F BB 0881 1961 : PUSHR #*M<R2,R3,R4,R5,R7,R8> ; Save regs
0885 1962 :
0885 1963 : MOVL R2,R7 ; Save channel number
55 57 52 D0 0888 1964 : MOVL RCB$L_ACP UCB(R0),R5 ; Get the ACP's UCB
14 A0 D0 088C 1965 : MOVZWL #MSG$_PATRLOST,R8 ; Setup mailbox message code
58 36 3C 088F 1966 : MOVL #6,R2 ; No. of bytes to be entered
52 06 D0 0892 1967 : BSBW NET$SEND_MBX ; Setup the message
02DC 30 0895 1968 : BLBC R0,30$ ; Br on error -- ignore it
09 50 E9 0898 1969 : MOVL PCB$L_PID(R4),(R3)+ ; Enter the PID
83 60 A4 D0 089C 1970 : MOVW R7,(R3)+ ; Enter channel
83 57 B0 089F 1971 : JSB @($P)+ ; Send the message
9E 16 08A1 1972 :
01BC 8F BA 08A1 1973 30$: POPR #*M<R2,R3,R4,R5,R7,R8> ; Restore regs
08A5 1974 50$:
08A5 1975 :
08A5 1976 : If the unit is busy then it must be a bug sinc NET$STAR!IO never
08A5 1977 : allows an I/O queue to build on the UCB
08A5 1978 :
08A5 1979 :
01 64 A5 08 E0 08A5 1980 : BBS #UCB$V_BSY,UCB$W_STS(R5),100$ ; Done if UCB is not busy
05 08AA 1981 : RSB ; Done
08AB 1982 :
08AB 1983 100$: BUG_CHECK NETNOSTATE,FATAL ; Our UCB assumptions are wrong
08AF 1984 :
```

```
08AF 1986 .SBTTL NET$PURG_RUN - Cleanup XWB to exit RUN state
08AF 1987 :
08AF 1988 :
08AF 1989 : The receiver and transmitter are run-down on both the DATA and INT/LS
08AF 1990 : LSB's.
08AF 1991 :
08AF 1992 : It is assumed that this routine is called as a result of a call from one
08AF 1993 : of the state transition action routines and that there will be a state
08AF 1994 : transition out of the RUN state as the event processing is completed. This
08AF 1995 : is because certain processing -- such as the setting and clearing of XWB
08AF 1996 : flags -- is assumed to be done as part of the state transition processing
08AF 1997 : and is therefore done by this routine.
08AF 1998 :
08AF 1999 :
08AF 2000 : INPUTS: R5 XWB address; low bit set if no XWB
08AF 2001 : R0 Scratch
08AF 2002 :
08AF 2003 : OUTPUTS: R0 Garbage
08AF 2004 :
08AF 2005 : All other registers are preserved
08AF 2006 :
08AF 2007 :
08AF 2008 NET$PURG_RUN::
08AF 2009 PUSH R5 ; Leave the RUN state
08AF 2010 PUSH R6 ; Save regs
08AF 2011 BBS #XWB$V_STS_CON,XWB$W_STS(R5),20$ ; If BS, not in RUN format
08AF 2012 BSBB DRAIN_XMT ; Drain the transmitter
08AF 2013 BSBW DRAIN_RCV ; Drain the receiver
08AF 2014
08AF 2015 20$: POP R5 ; Restore regs
08AF 2016 RSB
08AF 2017
08AF 2018
08AF 2019 DRAIN_XMT: ; Drain the xmitter
08AF 2020 :
08AF 2021 : All transmit CXB's are detached and eventually deallocated.
08AF 2022 : All transmit IRP's are sent to I/O Post with disconnect status.
08AF 2023 : The LSB transmitter state variables are updated to reflect an
08AF 2024 : idle transmitter.
08AF 2025 :
08AF 2026 :
08AF 2027 : Inputs: R8,R7 Scratch
08AF 2028 : R5 XWB address
08AF 2029 : R4-R0 Scratch
08AF 2030 :
08AF 2031 : Outputs: R8,R4-R0 garbage.
08AF 2032 :
08AF 2033 : All other registers are preserved.
08AF 2034 :
08AF 2035 :
08AF 2036 BSBW NET$MAP_R_REASON ; Map disconnect reason to status
08AF 2037 MOVZWL REASON_W_SS(R0),R0 ; Get proper I/O status code
08AF 2038 CLRL R1 ; IOSB second longword
08AF 2039 MOVAB XWB$T_LI(R5),R8 ; Get the LS/INT LSB
08AF 2040 BSBB 10$ ; Do it
08AF 2041 MOVAB XWB$T_DT(R5),R8 ; Get the DATA LSB
08AF 2042 BSBB 10$ ; Do it
```

01DE 8F BB 08AF 2009
05 OE A5 04 E0 08B3 2010
08 10 08B3 2011
0064 30 08B8 2012
08BA 2013
08BD 2014
01DE 8F BA 08BD 2015
05 08C1 2016
08C2 2017
08C2 2018
08C2 2019
08C2 2020
08C2 2021
08C2 2022
08C2 2023
08C2 2024
08C2 2025
08C2 2026
08C2 2027
08C2 2028
08C2 2029
08C2 2030
08C2 2031
08C2 2032
08C2 2033
08C2 2034
08C2 2035
50 F9FB 30 08C2 2036
02 A0 3C 08C5 2037
51 D4 08C9 2038
58 00D4 C5 9E 08CB 2039
11 10 08D0 2040
58 00A4 C5 9E 08D2 2041
0A 10 08D7 2042

```
08D9 2043
08D9 2044
08D9 2045
08D9 2046
08D9 2047
08D9 2048
08D9 2049
08D9 2050
08D9 2051
54 0D A8 9A 08D9 2052 MOVZBL LSB$B_X_CXBACT(R8),R4 ; Number of active Xmt CXB's
      03 13 08DD 2053 BEQL 5$ ; If EQL then none
      F71E' 30 08DF 2054 BSBW NET$ACK_XMT_SEGS ; "ACK" each segment release CXB's
      05 08E2 2055 5$: RSB ; Done
      08E3 2056
      08E3 2057
      7E 50 7D 08E3 2058 10$: MOVQ R0,-(SP) ; Save IOSB image
      08E6 2059
      08E6 2060
      08E6 2061
      08E6 2062
      08E6 2063
02 A8 68 B0 08E6 2064 MOVW LSB$W_LUX(R8),LSB$W_LNX(R8) ; Pretend we've sent all packets
06 A8 68 B0 08EA 2065 MOVW LSB$W_LUX(R8),LSB$W_HAR(R8) ; Pretend all packets were ACK'd
08 A8 68 B0 08EE 2066 MOVW LSB$W_LUX(R8),LSB$W_HAA(R8) ; No further ACK's expected
04 A8 68 B0 08F2 2067 MOVW LSB$W_LUX(R8),LSB$W_HXS(R8) ; No further packets to send
      08F6 2068
      08F6 2069
      08F6 2070
      08F6 2071
      08F6 2072
      08F6 2073
      08F6 2074
51 14 A8 9E 08F6 2075 MOVAB LSB$L_X_IRP(R8),R1 ; Get spent IRP listhead
      0D 11 08FA 2076 BRB 40$
      10 A8 D4 08FC 2077 20$: CLRL LSB$L_X_PND(R8) ; Detach pending IRP list
      61 50 D0 08FF 2078 MOVL R0,(RT) ; Attach it to end of spent IRP list
      51 50 D0 0902 2079 30$: MOVL R0,R1 ; Update last IRP pointer
38 A0 6E 7D 0905 2080 MOVQ (SP),IRP$L_IOST1(R0) ; Overwrite status
      50 61 D0 0909 2081 40$: MOVL (R1),R0 ; Get next IRP
      54 12 090C 2082 BNEQ 30$ ; If NEQ, IRP was found
50 10 A8 D0 090E 2083 MOVL LSB$L_X_PND(R8),R0 ; Get pending IRP list
      E8 12 0912 2084 BNEQ 20$ ; If NEQ, not empty
53 14 A8 D0 0914 2085 MOVL LSB$L_X_IRP(R8),R3 ; Get first IRP
      03 13 0918 2086 BEQL 100$ ; If EQL, none
      F6E3' 30 091A 2087 BSBW NET$XMT_DONE ; Complete all Xmt IRPs
      091D 2088
      50 8E 7D 091D 2089 100$: MOVQ (SP)+,R0 ; Restore stack and R0
      05 0920 2090 RSB ; Done
      0921 2091
      0921 2092
      0921 2093 DRAIN_RCV: ; Drain the receiver
      0921 2094
      0921 2095
      0921 2096
      0921 2097
      0921 2098
      0921 2099
      ; All receive CXB's are detached and deallocated.
      ; All receive IRP's are sent to I/O Post with disconnect status.
      ; For each LSB, LSB$B_R_CXBQUO is zeroed to prevent further CXB's
      ; from being received.
```

```
0921 2100
0921 2101
0921 2102
0921 2103
0921 2104
0921 2105
0921 2106
0921 2107
0921 2108
0921 2109
0921 2110
0921 2111
0921 2112
50 02 A0 3C 0924 2113
58 00D4 C5 9E 0928 2114
58 00A4 C5 9E 092A 2115
7E 50 7D 092F 2116
0931 2117
0936 2118 10$:
0939 2119
0939 2120
0939 2121
0939 2122
0939 2123
29 A8 94 0939 2124
28 A8 94 093C 2125
50 20 A8 D0 093F 2126
0E 13 0943 2127
20 A8 D4 0945 2128
0948 2129
10 A0 DD 0948 2130 30$:
0400 30 094B 2131
50 8ED0 094E 2132
F5 12 0951 2133
0951 2134
0953 2135 40$:
0953 2136
0953 2137
0953 2138
0953 2139
53 1C A8 D0 0953 2140
0C 13 0957 2141
32 A3 B4 0959 2142
38 A3 6E 7D 095C 2143
F69D' 30 0960 2144
EE 11 0963 2145
0965 2146
50 8E 7D 0965 2147 50$:
05 0968 2148
0969 2149

Inputs: R8 Scratch
R5 XWB address
R3 Scratch
R1-R0 Scratch

Outputs: R8,R3,R1,R0 garbage.

All other registers are preserved.

BSBW NET$MAP_R_REASON ; Map disconnect reason to status
MOVZWL REASON_Q_SS(R0),R0 ; Get proper I/O status code
CLRL R1 ; IOSB second longword
MOVAB XWB$T_LI(R5),R8 ; Get the LS/INT LSB
BSBB 10$ ; Do it
MOVAB XWB$T_DT(R5),R8 ; Get the DATA LSB
MOVQ R0,-(SP) ; Save IOSB image

Drain Receive CXB List

CLRB LSB$B_R_CXBQUO(R8) ; Prevent further receives
CLRB LSB$B_R_CXBCNT(R8) ; Zero the CXB in use count
MOVL LSB$L_R_CXB(R8),R0 ; Get first CXB in list
BEQL 40$ ; If EQL then none
CLRL LSB$L_R_CXB(R8) ; Detach entire CXB chain from LSB

PUSHL CXB$L_LINK(R0) ; Save ptr to next CXB
RSBW NET$DEALLOCATE ; Deallocate block in R0
POPL R0 ; Get the next CXB

BNEQ 30$ ; If NEQ then loop, else no CXB

Complete all Rcv IRP's with mapped disconnect status code

MOVL LSB$L_R_IRP(R8),R3 ; Get next Rcv IRP
BEQL 50$ ; If EQL then none
CLRW IRP$W_BCNT(R3) ; No bytes xferred
MOVQ (SP),IRP$L_IOST1(R3) ; Setup I/O status
BSBW NET$RCV_DONE ; Complete the receive
BRB 40$ ; Loop

MOVQ (SP)+,R0 ; Restore regs
RSB ; Done
```

```
0969 2151 .SBTTL NET$ACP_COMM - Entry for ACP communication
0969 2152 :++
0969 2153 :
0969 2154 : This routine is called by the ACP for change of status notification
0969 2155 : including process exit, logical-link 'ownership' changes, and datalink
0969 2156 : transitions.
0969 2157 :
0969 2158 :
0969 2159 : CALLING SEQUENCE:
0969 2160 :
0969 2161 :     JSB  @ACRBSL_INTD+VEC$SL_START  at IPL 0
0969 2162 :
0969 2163 : INPUTS: R5  NET UCB address.
0969 2164 :         R4-R1 Function specific -- see individual action routine preambles
0969 2165 :         R0  Function code as follows:
0969 2166 :
0969 2167 :     NETUPDS$CONNECT - Pass NCB to Declared Name mailbox
0969 2168 :     NETUPDS$PROCRE  - Process created to received connect
0969 2169 :     NETUPDS$ABORT   - Process couldn't start
0969 2170 :     NETUPDS$EXIT    - Started process is exiting
0969 2171 :
0969 2172 :     NETUPDS$DLL_ON  - Datalink has come online - post a receive
0969 2173 :     NETUPDS$DLL_DLE - Datalink online for service fcts
0969 2174 :     NETUPDS$REACT_RCV - Reactivate Datalink receiver
0969 2175 :     NETUPDS$SEND_HELLO - Force datalink to send a hello message
0969 2176 :
0969 2177 :     NETUPDS$CRELNK  - Create a logical-link control structure
0969 2178 :     NETUPDS$DSCLNK  - Graceful disconnect of single link
0969 2179 :     NETUPDS$ABOLNK  - Force immediate disconnect of all links
0969 2180 :
0969 2181 :     NETUPDS$BRDCST  - Broadcast mailbox message
0969 2182 :     NETUPDS$REPLY   - Reply to associated mailbox
0969 2183 :
0969 2184 : OUTPUTS: R0 Status
0969 2185 :
0969 2186 :     All other registers are preserved.
0969 2187 :
0969 2188 :--
0000000C 0969 2189 :
00000010 0969 2190 :     R3_OFF = 4*3
00000014 0969 2191 :     R4_OFF = 4*4
0969 2192 :     R5_OFF = 4*5
0969 2193 :
0969 2194 :     .ENABL  LSB
0969 2195 : NET$ACP_COMM::
0969 2196 :     SETIPL  UCB$B_FIPL(R5) ; ACP entry point
0969 2197 :                                     ; Raise IPL to synch access to structures
096D 2198 :
07FF 8F  BB 096D 2199 :     PUSHR  #*M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10> ; Save regs
0971 2199 :
0971 2200 :     MOVL   SP,R10 ; Save ptr to saved R0
0974 2201 :     BSBB   20$ ; Dispatch on fct code
0976 2202 :     MOVL   R0,(SP) ; Overlay return code
0979 2203 :
0979 2204 :     POPR   #*M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10> ; Restore regs
097D 2205 :
097D 2206 :     SETIPL #0 ; Restore IPL
0980 2207 :     RSB
```

```
0981 2208
0981 2209 20$: $DISPATCH R0,TYPE=B,- ; Case on function code
0981 2210 <-
0981 2211 <NETUPD$_CONNECT, DECLARE>,- ; Pass NCB to Declared Name mailbox
0981 2212 <NETUPD$_PROCRE, PROCRE>,- ; Process created to rcv connect
0981 2213 <NETUPD$_ABORT, ABORT>,- ; Abort single link for given process
0981 2214 <NETUPD$_EXIT, EXIT>,- ; Started process is exiting
0981 2215 -
0981 2216 <NETUPD$_CRELNK, CRE_LNK>,- ; Create a logical-link
0981 2217 <NETUPD$_DSCLNK, DIST_ONE>,- ; Disconnect single logical-link
0981 2218 <NETUPD$_ABOLNK, ABORT_ALL>,- ; Abort all logical-links
0981 2219 -
0981 2220 <NETUPD$_BRDCST, BRDCST>,- ; Broadcast mailbox message
0981 2221 <NETUPD$_REPLY, REPLY>,- ; Send general mailbox message
0981 2222 <NETUPD$_DLL_ON, DLLTRN>,- ; Datalink made into 'on' state
0981 2223 -
0981 2224 >
018F 31 099B 2225 > BRW UNKNOWN ; Let lowest level handle this
099E 2226
099E 2227 ;+
099E 2228 : PROCRE - Process started due to CI received
099E 2229 :
099E 2230 : INPUTS: R5 NET UCB address.
099E 2231 : R4 Scratch
099E 2232 : R3 Local link number.
099E 2233 : R2 Scratch
099E 2234 : R1 PID of process
099E 2235 :
099E 2236 :-
50 D4 099E 2237 PROCRE: CLRL R0 ; Setup for 'no PID' match
009D 30 09A0 2238 BSBW 200$ ; Get XWB
11 12 09A3 2239 BNEQ 40$ ; Done if NEQ
34 A5 51 D0 09A5 2240 MOVL R1,XWBSL_PID(R5) ; Set PID of process allowed
0B 11 09A9 2241 ; to complete the connect
09AB 2242 BRB 40$ ; Done
09AB 2243
09AB 2244 ;+
09AB 2245 : ABORT - Abort single logical-link for a given process
09AB 2246 :
09AB 2247 : INPUTS: R5 NET UCB address.
09AB 2248 : R4 Scratch
09AB 2249 : R3 Local link number.
09AB 2250 : R2 Disconnect reason code
09AB 2251 : R1 PID of process (zero if process not started)
09AB 2252 :
09AB 2253 :-
50 51 D0 09AB 2254 ABORT: MOVL R1,R0 ; Setup PID (could be zero)
008F 30 09AE 2255 BSBW 200$ ; Get XWB
03 12 09B1 2256 BNEQ 40$ ; Done if NEQ
0067 30 09B3 2257 BSBW 180$ ; Enter DIS state
50 01 D0 09B6 2258 40$: MOVL S^#SS$_NORMAL,R0 ; Report success
05 05 09B9 2259 RSB ; Done
09BA 2260
09BA 2261 ;+
09BA 2262 : EXIT - A formerly started process has exited
09BA 2263 :
09BA 2264 : INPUTS: R5 NET UCB address
```

```
09BA 2265 : R4 Scratch
09BA 2266 : R3 Scratch
09BA 2267 : R2 Disconnect reason code
09BA 2268 : R1 PID of process
09BA 2269 :
09BA 2270 :
55 34 A5 DO 09BA 2271 EXIT: MOVL UCB$$_VCB(R5),R5 ; Get RCB
1B 13 09BE 2272 BEQL 80$ ; Br if not mounted
55 24 A5 DO 09C0 2273 MOVL RCB$$_PTR_LTB(R5),R5 ; Get LTB
15 13 09C4 2274 BEQL 80$ ; Br if its not there
55 E0 A5 DE 09C6 2275 MOVAL -XWB$$_LINK - ;
09CA 2276 +LTB$$_XWB(R5),R5 ; Setup for scan
55 2C A5 DO 09CA 2277 60$: MOVL XWB$$_LINK(R5),R5 ; Get next XWB
0B 13 09CE 2278 BEQL 80$ ; If EQL then end of list
50 51 DO 09D0 2279 MOVL R1,R0 ; Copy process PID
73 10 09D3 2280 BSBB 210$ ; Check process access to XWB via PID
F3 12 09D5 2281 BNEQ 60$ ; If NEQ then something wrong
44 10 09D7 2282 BSBB 180$ ; Disconnect the link
EF 11 09D9 2283 BRB 60$ ; Continue
50 01 DO 09DB 2284 80$: MOVL S^#SS$_NORMAL,R0 ; Success
05 09DE 2285 RSB ; Done
09DF 2286
09DF 2287
09DF 2288 :+
09DF 2289 : CRE_LNK - Create a single logical-link
09DF 2290 :
09DF 2291 : INPUTS: R5 NET UCB address.
09DF 2292 : R4 Scratch
09DF 2293 : R3 Logical-link's remote node address
09DF 2294 : R2 Scratch
09DF 2295 : R1 PID of process allowed to access link
09DF 2296 :
09DF 2297 : OUTPUTS: R0 XWB address, high bit clear => failure code
09DF 2298 :
09DF 2299 : CRE_LNK:
09DF 2300 : BSBW NET$CREATE_XWB ; Create single logical-link
09E2 2301 BLBC R0,10$ ; Create the structure
34 A5 07 50 DO 09E5 2302 MOVL R1,XWB$$_PID(R5) ; If LBC, failed
50 55 DO 09E9 2303 MOVL R5,R0 ; Setup PID
05 09EC 2304 10$: RSB ; Setup XWB address
09ED 2305 ; Done
09ED 2306 :+
09ED 2307 : DISC_ONE - Disconnect a single logical-link
09ED 2308 :
09ED 2309 : INPUTS: R5 NET UCB address.
09ED 2310 : R4 Scratch
09ED 2311 : R3 Local link number.
09ED 2312 : R2 Disconnect reason code
09ED 2313 : R1 Logical-link's remote node address
09ED 2314 :
09ED 2315 : DISC_ONE:
09ED 2316 : BSBW XWB_LOCLNK ; Disconnect single logical-link
50 14 3C 09F0 2317 MOVZWL S^#SS$_BADPARAM,R0 ; Find the logical-link XWB
11 55 E8 09F3 2318 BLBS R5,120$ ; Assume no such link exists
3A A5 B5 09F6 2319 TSTW XWB$$_REMNODE(R5) ; If LBS then XWB was not found
06 13 09F9 2320 BEQL 100$ ; Remote node 0?
3A A5 51 B1 09FB 2321 CMPW R1,XWB$$_REMNODE(R5) ; If so, ignore node check
; Same remote node ?
```

```
06 12 09FF 2322      BNEQ 120$      ; If not, return error
0019 30 0A01 2323 100$: BSBW 180$      ; Disconnect the link
50 01 D0 0A04 2324      MOVL S^#SS$ _NORMAL,R0 ; Success
05 05 0A07 2325 120$: RSB
0A08 2326
0A08 2327 ;+
0A08 2328 ; ABORT_ALL - Abort all logical-links
0A08 2329 ;
0A08 2330 ; INPUTS. R5 NET UCB address
0A08 2331 ; R4 Scratch
0A08 2332 ; R3 Scratch
0A08 2333 ; R2 Scratch
0A08 2334 ; R1 Ptr to LTB
0A08 2335 ;
0A08 2336 ;
0A08 2337 ABORT_ALL:
55 E0 A1 DE 0A08 2338 MOVAL -XWBS$ _LINK - ; Abort all logical-links
0A0C 2339 +LTB$ _XWB(R1),R5 ; Prepare for scan
55 2C A5 D0 0A0C 2340 140$: MOVL XWBS$ _LINK(R5),R5 ; Get next XWB
07 13 0A10 2341 BEQL 160$ ; If NEQ then got one
52 08 B0 0A12 2342 MOVW #NET$C_DR_THIRD,R2 ; Reason is "third party abort"
06 10 0A15 2343 BSBW 180$ ; Mark link to be broken
F3 11 0A17 2344 BRB 140$ ; Loop
50 01 D0 0A19 2345 160$: MOVL S^#SS$ _NORMAL,R0 ; Success
05 05 0A1C 2346 RSB
0A1D 2347
0A1D 2348
0A1D 2349 ;
0A1D 2350 ; Disconnect the link
0A1D 2351 ;
0A1D 2352 180$: CMPW XWBS$ X REASON(R5),- ; Remote reason been setup yet?
0064 8F B1 0A20 2353 #NET$C_DR_INVALID ;
04 12 0A23 2354 BNEQ 190$ ; If NEQ then yes
46 A5 52 B0 0A25 2355 MOVW R2,XWBS$ X REASON(R5) ; Enter disconnect reason
44 A5 B1 0A29 2356 190$: CMPW XWBS$ R REASON(R5),- ; Local reason been setup yet?
0064 8F 0A2C 2357 #NET$C_DR_INVALID ;
04 12 0A2F 2358 BNEQ 195$ ; If NEQ then yes
44 A5 52 B0 0A31 2359 MOVW R2,XWBS$ R REASON(R5) ;
F5C8 30 0A35 2360 195$: BSBW NET$MARK_LINK ; Mark the link to be broken
50 6A 7D 0A38 2361 MOVQ (R10),R0 ; Restore R0,R1,R2
52 08 AA D0 0A3B 2362 MOVL 8(R10),R2 ;
05 05 0A3F 2363 RSB ; Done
0A40 2364 ;
0A40 2365 ; Find XWB, verify access rights by PID
0A40 2366 ;
0254 30 0A40 2367 200$: BSBW XWB_LOCLNK ; Find XWB via local link number
01 95 0A43 2368 TSTB #1 ; Clear Z-bit, assuming error
0A 55 E8 0A45 2369 BLBS R5,220$ ; If LBS then no XWB
34 A5 50 D1 0A48 2370 210$: CMPL R0,XWBS$ _PID(R5) ; Is the process the owner ?
04 12 0A4C 2371 BNEQ 220$ ; If NEQ then no
03 91 0A4E 2372 CMPB #XWBS$C_STA_CIR,- ;
1E A5 0A50 2373 XWBS$B_STA(R5) ; Verify state
05 05 0A52 2374 220$: RSB
0A53 2375
0A53 2376
0A53 2377 ;+
0A53 2378 ; BRDCST - Broadcast a mailbox message
```

```
0A53 2379 :  
0A53 2380 : INPUTS: R5 NET UCB address  
0A53 2381 : R4 Ptr to mailbox msg text  
0A53 2382 : R3 Associated mailbox mask (0 if broadcast to all mailboxes)  
0A53 2383 : R2 Mailbox msg code  
0A53 2384 : R1 Scratch  
0A53 2385 :  
0A53 2386 :  
0A53 2387 : BRDCST: ; Broadcast mailbox message  
0A53 2388 :  
0A53 2389 : ; & Code to set up R3 here will move to NETACP, eventually  
0A53 2390 :  
58 F791 CF 9E 0A53 2391 MOVAB MBX_TABLE,R8 ; Point to filter mapping table  
53 88 D0 0A58 2392 300$: MOVL (R8)+,R3 ; Get next mask  
05 13 0A5B 2393 BEQL 320$ ; If EQL at end of table - take the msg  
52 88 B1 0A5D 2394 CMPW (R8)+,R2 ; Is this the msg being sent?  
F6 12 0A60 2395 BNEQ 300$ ; If NEQ no - loop; else, R3 has bit  
58 52 D0 0A62 2396 320$: MOVL R2,R8 ; Transfer msg type code  
00 DD 0A65 2397 PUSHL #0 ; Assume no message text  
57 5E D0 0A67 2398 MOVL SP,R7 ; Point to it  
54 D5 0A6A 2399 TSTL R4 ; Any message text?  
52 64 9A 0A6C 2400 BEQL 400$ ; If EQL no, goto end of loop  
52 52 D6 0A71 2401 MOVZBL (R4),R2 ; Get count field value  
57 54 D0 0A73 2402 INCL R2 ; Inc to get total string size  
1C 11 0A76 2403 MOVL R4,R7 ; Setup stable string pointer  
0A78 2404 BRB 400$ ; Jump to end of loop  
0A78 2405  
53 D5 0A78 2406 340$: TSTL R3 ; Will everyone take this message?  
06 13 0A7A 2407 BEQL 360$ ; If EQL yes  
44 A5 53 D3 0A7C 2408 BITL R3,UCB$_DEVDEPEND(R5) ; Can this UCB take this message?  
12 13 CA80 2409 BEQL 400$ ; If EQL no - don't even try to send  
2C BB 0A82 2410 360$: PUSHR #*M<R2,R3,R5> ; Save regs  
00EA 30 0A84 2411 BSBW NET$SEND-MBX ; Call co-routine to setup the message  
08 50 E9 0A87 2412 BLBC R0,380$ ; If LBC then error  
51 57 D0 0A8A 2413 MOVL R7,R1 ; Get message pointer  
02DB 30 0A8D 2414 BSBW NET$MOV_CSTR ; Move the string with count field  
9E 16 0A90 2415 JSB @ (SP)+ ; Complete the message  
55 30 A5 D0 0A92 2416 380$: POPR #*M<R2,R3,R5> ; Recover regs  
DE 12 0A94 2417 400$: MOVL UCB$_LINK(R5),R5 ; Get next UCB  
8E D5 0A9A 2418 BNEQ 340$ ; If NEQ then got one  
50 01 D0 0A9C 2419 TSTL (SP)+ ; Fix the stack  
05 0A9F 2420 MOVL S^#SS$_NORMAL,R0 ; Exit with success  
0AA0 2421 RSB  
0AA0 2422  
0AA0 2423  
0AA0 2424 :+  
0AA0 2425 : REPLY - Send general message to associated mailbox  
0AA0 2426 :  
0AA0 2427 : INPUTS: R5 NET UCB address  
0AA0 2428 : R4 Ptr to mailbox msg text  
0AA0 2429 : R3 & Associated mailbox mask if NETUPD$_BRDCST (0 if broadcast all)  
0AA0 2430 : R2 Mailbox msg code  
0AA0 2431 : R1 Scratch  
0AA0 2432 :  
0AA0 2433 :  
58 52 D0 0AA0 2434 REPLY: MOVL R2,R8 ; Get mailbox message code  
13 11 0AA3 2435 BRB 500$ ; Continue in common
```

```
0AAS 2436
0AAS 2437
0AAS 2438 :+ DECLARE - Pass NCB to Declared-name mailbox
CAAS 2439
0AAS 2440 : INPUTS: R5 NET UCB address
0AAS 2441 : R4 Ptr to NCB counted string
0AAS 2442 : R3 Scratch
0AAS 2443 : R2 Scratch
0AAS 2444 : R1 Scratch
0AAS 2445
0AAS 2446 :-
0AAS 2447 DECLARE:
01EF 30 0AAS 2448 BSBW XWB_LOCLNK : Pass NCB to declare-object mailbox
29 55 E8 0AAB 2449 BLBS R5,560$ : Find link's XWB
03 91 0AAB 2450 CMPB #XWB$C_STA CIR,- : Br if no XWB
1E A5 0AAD 2451 XWB$B_STA(R5) : Must be in CIR state
23 12 0AAF 2452 BNEQ 560$ : If not then cannot redirect connect
34 A5 51 D0 0AB1 2453 MOVL R1,XWB$C_PID(R5) : Set PID of process
58 32 3C 0ABS 2454 MOVZWL #MSG$ CONNECT,R8 : Setup mailbox message type
54 10 AA 7D 0AB8 2455 500$: MOVQ R4,OFF(R10),R4 : Get mbx message and UCB addresses
52 54 D0 0ABC 2456 MOVL R4,R2 : Copy msg pointer
03 13 0ABF 2457 BEQL 520$ : If EQL then no text
52 62 9A 0AC1 2458 MOVZBL (R2),R2 : Set count of bytes to be sent
00AA 30 0AC4 2459 520$: BSBW NET$SEND_MBX : Prepare to send mailbox message
0D 50 E9 0AC7 2460 BLBC R0,580$ : Br on error
51 54 D0 0ACA 2461 MOVL R4,R1 : Copy NCB pointer
03 13 0ACD 2462 BEQL 540$ : Skip if null
0299 30 0ACF 2463 BSBW NET$MOV_CSTR : Move counted string into buffer
9E 16 0AD2 2464 540$: JSB @ (SP)+ : Complete writing mailbox
50 01 D0 0AD4 2465 560$: MOVL S^#SS$_NORMAL,R0 : Success
05 05 0AD7 2466 580$: RSB : Done
0AD8 2467
0AD8 2468
0AD8 2469 :+
0AD8 2470 : DLLTRN - Datalink state transition
0AD8 2471 :
0AD8 2472 : INPUTS: R5 NET UCB address
0AD8 2473 : R4 Scratch
0AD8 2474 : R3 Scratch
0AD8 2475 : R2 Scratch
0AD8 2476 : R1 Ptr to datalink's LPD
0AD8 2477 :-
52 34 A5 D0 0AD8 2479 DLLTRN: MOVL UCB$C_VCB(R5),R2 : Get RCB
01 91 0ADC 2480 CMPB #LPD$C_LOC_INX,- : Is this the local LPD
20 A1 0ADE 2481 LPD$B_PTH_INX(R1)
4B 12 0AE0 2482 BNEQ UNK..OWN : If not, branch
3F BB 0AE2 2483 :
53 55 D0 0AE4 2484 PUSHR #^M<R0,R1,R2,R3,R4,R5> : Save regs
30 A2 D0 0AE7 2485 MOVL R5,R3 : Copy UCB address
02 E0 0AEB 2486 MOVL RCB$C_PTR_TQE(R2),R5 : Get TQE
3B 0B A5 E0 0AEB 2487 BBS #TQE$V_REPEAT,- : Br if timer is in use
05 90 0AF0 2488 TQE$B_RQTYPE(R5),600$ :
0B A5 0AF0 2489 MOVAB #TQE$C_SSREPT,- : Set for system subroutine repeat
0000 CF 9E 0AF2 2490 TQE$B_RQTYPE(R5)
0C A5 0AF4 2491 MOVAB W^NET$TIMER,- : Set timer handler address
0AF8 2492 TQE$C_FPC(R5)
```

NETDRVSES
V04-000

E 16
- DECnet Session Control Module for NETD 16-SEP-1984 01:32:10 VAX/VMS Macro V04-00
NET\$ACP_COMM - Entry for ACP communicati 5-SEP-1984 02:20:26 [NETACP.SRC]NETDRVSES.MAR;1 Page 57
(50)

```

      14 A5 53 D0 0AFA 2493      MOVL R3,TQESL,FR4(R5)      ; Save UCB address
00000000 00989680 8F 7D 0AFE 2494      MOVQ #10*1000*1000 -      ; 1 tick = 1 sec
      20 A5 0B08 2495      TQESQ DELTA(R5)
      50 F60A CF 9E 0B0A 2496      MOVAB W^NET$GL_OFF_DPTFLG,R0      ; Get address of offset to DPT$B_FLAGS
      50 60 60 C0 0B0F 2497      ADDL (R0),R0      ; Make it an address
      60 04 88 0B12 2498      BISB #DPT$M_NOUNLOAD,(R0)      ; Prevent reload of driver
50 00000000'GF 7D 0B15 2499      MOVQ G^EXESQ SYSTIME,R0      ; Set time of first tick
      00000000'GF 16 0B22 2501      JSB G^EXESINSTIME      ; Lower IPL to that of timer service
      3F BA 0B28 2502      ENBINT      ; Insert into queue
      F4D0' 30 0B2B 2503 600$: POPR #^M<R0,R1,R2,R3,R4,R5>      ; Restore IPL
      05 0B2D 2504      :
      0B2D 2505 UP VQWN:      :
      0B2D 2506      BSBW TR$UPDATE      ; Fct code is in R0
      0B30 2507      RSB      ; Exit with status in R0
      0B31 2508      :
      0B31 2509      :
      0B31 2510      .DSABL LSB
```

```
0B31 2512 .SBTTL NET$SEND_CS_MBX - Send counted string to mailbox
0B31 2513 :+
0B31 2514 :
0B31 2515 : A mailbox message is built and sent to the mailbox associated with the UCB
0B31 2516 : associated with the XWB. The counted string pointed to by R1 is appended to
0B31 2517 : the end of the mailbox message. R2 contains the assumed total count of the
0B31 2518 : string and may be zero. If there is no mailbox then the routine is assumed
0B31 2519 : implicitly successful.
0B31 2520 :
0B31 2521 :
0B31 2522 INPUTS: R8 Mailbox message type code
0B31 2523 R5 XWB address
0B31 2524 R2 Assumed total length of string (low byte only)
0B31 2525 R1 Address of count field of string
0B31 2526 :
0B31 2527 OUTPUTS: R2 Zero
0B31 2528 R1 Garbage
0B31 2529 R0 SS$_NORMAL if mailbox successfully written
0B31 2530 SS$_NOMBX!1 if no associated mailbox or no UCB
0B31 2531 Zero if (R1)+1 NEQ R2 or R2 GTRU 17
0B31 2532 Also see NET$SEND_MBX for R0 error codes
0B31 2533 :
0B31 2534 :
0B31 2535 : All other registers are preserved
0B31 2536 :
0B31 2537 :-
0B31 2538 NET$SEND_CS_MBX::
0B31 2539 PUSH R #^M<R1,R2,R3,R4,R5> ; Save regs
0B33 2540 :
0B33 2541 TSTL R2 ; Any bytes in string ?
0B35 2542 BNEQ 10$ ; If NEQ yes, else can't trust R1
0B37 2543 MOVAB B^50$,(SP) ; Setup null string ptr
0B38 2544 BRB 20$ ; Continue
0B3D 2545 10$: CLRL R0 ; Assume string error
0B3F 2546 CMPL R2,#17 ; Is count within range
0B42 2547 BGTRU 40$ ; If not, branch
0B44 2548 SUBB3 (R1),R2,R3 ; Check count field consistency
0B48 2549 DECB R3 ; Account for count field itself
0B4A 2550 BNEQ 40$ ; Inconsistent if NEQ
0B4C 2551 20$: MOVZWL #SS$_NOMBX!1,R0 ; Assume no UCB or mailbox
0B51 2552 MOVL XWB$_ORGUCB(R5),R5 ; Get UCB
0B55 2553 BEQL 40$ ; If none, done
0B57 2554 TSTL UCB$_AMB(R5) ; Is there a mailbox ?
0B5A 2555 BEQL 40$ ; If not, branch
0B5C 2556 BSBW NET$SEND_MBX ; Build header (co-routine)
0B5F 2557 BLBC R0,40$ ; Br on error
0B62 2558 MOVL 4(SP),R1 ; Get string address (note stack)
0B66 2559 BSBW NET$MOV_CSTR ; Move string with count field
0B69 2560 JSB @ (SP)+ ; Close and send mbx message
0B6B 2561 :
0B6B 2562 40$: POPR #^M<R1,R2,R3,R4,R5> ; Recover regs
0B6D 2563 CLRL R2 ; String has been consumed
0B6F 2564 RSB
0B70 2565 :
0B70 2566 50$: .BYTE 0 ; Phony counted string for mailbox
```

```
0B71 2568 .SBTTL NET$SEND_MBX - Co-routine to send mailbox message
0B71 2569 :+
0B71 2570 :
0B71 2571 : The first time the routine is entered the associated mailbox is found, a
0B71 2572 : buffer is allocated for the message, and the mailbox header is built. When
0B71 2573 : the routine is re-entered, after a call-back to the co-routine, the message
0B71 2574 : is closed and sent to the mailbox.
0B71 2575 :
0B71 2576 : The original entry parameters are given below, the re-entry parameters are
0B71 2577 : given within the body of the code.
0B71 2578 :
0B71 2579 :
0B71 2580 : INPUTS: R8 Mailbox message type code
0B71 2581 : R5 UCB address
0B71 2582 : R3 Scratch
0B71 2583 : R2 Count of bytes co-routine will enter into message
0B71 2584 : R1 Scratch
0B71 2585 : R0 Scratch
0B71 2586 :
0B71 2587 : OUTPUTS: R3 Pointer to next byte in mailbox message to be filled
0B71 2588 : R2 Address of allocated buffer if R0=SS$_NORMAL
0B71 2589 : R1 Garbage
0B71 2590 : R0 SS$_NORMAL if successful
0B71 2591 : SS$_NOMBX if there's no associated mailbox
0B71 2592 :
0B71 2593 : see NET$ALONONPAGED for additional error status
0B71 2594 :
0B71 2595 : All other registers are preserved
0B71 2596 :
0B71 2597 :-
0B71 2598 NET$SEND_MBX::
0B71 2599 :
0B71 2600 :
0B71 2601 : Add 24 to the number of bytes the user will enter. This will
0B71 2602 : ensure that the allocated block is large enough for COM$DRVDEALMEM
0B71 2603 : to deallocate -- also creates space for:
0B71 2604 :
0B71 2605 : 12 bytes for standard buffer header
0B71 2606 : 2 bytes for mailbox msg type code
0B71 2607 : 2 bytes for mailbox unit number
0B71 2608 : 1 byte for count field for device name
0B71 2609 :
0B71 2610 :
0B71 2611 : ADDL #24,R2 ; Increase buffer size
50 52 18 C0 0B71 2612 : MOVL UCB$L_DDB(R5),R0 ; DDB pointer
51 28 A5 D0 0B74 2613 : MOVAB DDB$T_NAME(R0),R1 ; Get device name string ptr
51 14 A0 9E 0B78 2614 :
0B7C 2615 : PUSHL R1 ; Save device name string ptr
51 51 DD 0B7C 2616 : MOVZBL (R1),R1 ; Get string size
51 61 9A 0B7E 2617 : ADDL R2,R1 ; Add in remaining bytes
51 52 C0 0B81 2618 : BSBW NET$ALONONPAGED ; Get the buffer
01AD 30 0B84 2619 : POPL R1 ; Restore device name string ptr
51 8ED0 0B87 2620 :
0B8A 2621 : BLBS R0,10$ ; If LBS then okay
01 50 E8 0B8A 2622 : RSB ; Return with error status in R0
0B8D 2623 :
0B8E 2624 10$: ADDL3 #12,R2,R3 ; Get pointer to start of msg
53 52 0C C1 0B8E 2624
```

83	83	58	B0	OB92	2625	MOVW	R8,(R3)+	; Enter message type code
	54	A5	B0	OB95	2626	MOVW	UCB\$W_UNIT(R5),(R3)+	; Enter unit I.D.
		01CF	30	OB99	2627	BSBW	NET\$MOV_CSTR	; Move in device name with count field
	50	01	D0	OB9C	2628	MOVL	S^#SS\$_NORMAL,R0	; Indicate success
		9E	16	OB9F	2629	JSB	@(SP)+	; Call co-routine for more bytes
				OBA1	2630			; Note that R4 is unmodified
				OBA1	2631			
				OBA1	2632			
				OBA1	2633			
				OBA1	2634			
				OBA1	2635			
				OBA1	2636			
				OBA1	2637			
				OBA1	2638			
				OBA1	2639			
				OBA1	2640			
54	52	0C	C1	OBA1	2641	ADDL3	#12,R2,R4	; Get start of mbx message
	53	54	C2	OBA5	2642	SUBL	R4,R3	; Get length of mbx message
50	0274	8F	3C	OBA8	2643	MOVZWL	#SS\$_NOMBX,R0	; Assume no mailbox
55	60	A5	D0	OBA0	2644	MOVL	UCB\$_AMB(R5),R5	; Get mailbox
		06	13	OBB1	2645	BEQL	20\$	; If EQL then no mailbox
	00000000	'GF	16	OBB3	2646	JSB	G^EXE\$WRTMAILBOX	; Send message to mailbox
				OBB9	2647			
		50	DD	OBB9	2648	PUSHL	R0	; Save return status
50	54	0C	C3	OBBB	2649	SUBL3	#12,R4,R0	; Get buffer address
		018C	30	OBBF	2650	BSBW	NET\$DEALLOCATE	; Deallocate block in R0
		50	8ED0	OBC2	2651	POPL	R0	; Restore reg
				OBC5	2652			
			05	OBC5	2653			
				OBC6	2654	RSB		; Done

On coroutine return: R5 = UCB address
R3 = address of 1st byte past mbx msg
R2 = buffer address

On return to caller: R0 = EXE\$WRITEMBX status
R1-R5 are garbage

```

OBC6 2656 .SBTTL NET$CREATE_XWB - Create XWB for logical-link
OBC6 2657 :++
OBC6 2658 :
OBC6 2659 : An XWB (the logical-link control structure that will eventually be attached
OBC6 2660 : to an I/O channel (CB$SL_WIND field) is allocated and initialized, provided
OBC6 2661 : that the current maximum logical-link count is not exceeded. The current
OBC6 2662 : logical-link count is incremented.
OBC6 2663 :
OBC6 2664 : No local link address is assigned, and the XWB is not linked into the LTB.
OBC6 2665 :
OBC6 2666 :
OBC6 2667 : INPUTS:      R5      NET UCB Address
OBC6 2668 :           R3      Remote node address
OBC6 2669 :           R0      Scratch
OBC6 2670 :
OBC6 2671 : OUTPUTS:     R5      Address of XWB if successful, otherwise LBS
OBC6 2672 :           R0      Status
OBC6 2673 :
OBC6 2674 : All other registers are preserved
OBC6 2675 :
OBC6 2676 :--
OBC6 2677 NET$CREATE_XWB::                                : Get idle XWB
1E BB OBC6 2678 PUSH R1,R2,R3,R4>                          : Save regs to be used
OBC8 2679 :
OBC8 2680 :
OBC8 2681 : Make sure we are not over our limit (MOUNT = current links + 1).
OBC8 2682 :
OBC8 2683 :
50 OBC8 2684 MOVZWL #SS$_NOLINKS,R0                          : Assume failure
52 OBCD 2685 MOVL   UCB$SL_VCB(R5),R2                        : Point to RCB
OBCD 2686 BEQL    13$                                         : If EQL then no RCB
51 OBD3 2687 MOVZWL RCB$W_MCOUNT(R2),R1                    : Get current Mount Count
OBD7 2688 BEQL    13$                                         : If EQL, NETACP shutting down
58 A2 51 B1 OBD9 2689 CMPW   R1,RCB$W_MAX_LNK(R2)            : Is new link allowed?
OBD9 2690 BGTRU   12$                                         : If not, branch
51 O17C 2691 MOVZWL #XWB_C_LEN,R1                             : Get size of XWB
O13C 30 OBE4 2692 BSBW    NET$A_CONPGD_Z                     : Allocate the block and zero it
OBE7 2693 : to initialize most fields
51 52 DO OBE7 2694 MOVL    R2,R1                             : Save XWB pointer
52 34 A5 DO OBEA 2695 MOVL   UCB$SL_VCB(R5),R2                : Point to RCB
55 51 DO OBE7 2696 MOVL    R1,R5                             : Use standard XWB pointer
54 08 AE B0 OBF1 2697 MOVW   8(SP),R4                        : Get dst node address
10 50 E8 OBF5 2698 BLBS    R0,15$                             : Br if successful
OBF8 2699 12$: BUMP    W,RCB$W_CNT_XRE(R2)                    : Account for resource error
55 01 DO OC03 2700 13$: MOVL   #1,R5                          : Invalidate XWB ptr
14 11 OC06 2701 BRB     100$                                   : Done
OC08 2702 15$:
OC08 2703 :
OC08 2704 : Initialize the XWB and bump RCB mount count.
OC08 2705 :
OC08 2706 :
009E C2 15 10 OC08 2707 BSBB    INIT_XWB                          : Init XWB
54 A2 B1 OC0A 2708 CMPW    RCB$W_MCOUNT(R2),RCB$W_CNT_MLL(R2) : New max active links value?
04 1B OC10 2709 BLEQU   30$                                     : If LEQU then no
009E C2 B6 OC12 2710 INCW   RCB$W_CNT_MLL(R2)                  : Bump max active link count
OC16 2711 : (#links = MOUNT-1)
54 A2 B6 OC16 2712 30$: INCW   RCB$W_MCOUNT(R2)              : Account for new link
```

```
50 01 D0 0C19 2713 MOVL #1,R0 ; Success
1E 1E BA 0C1C 2714 100$: POPR #*M<R1,R2,R3,R4> ; Restore regs
05 05 0C1E 2715 RSB ; Done
0C1F 2716
0C1F 2717 INIT_XWB: ; Initialize XWB.
90 90 0C1F 2718 MOVB #NETSC_IPL, XWBSB_FIPL(R5) ; Setup fork IPL
90 90 0C23 2719 MOVB #DYN$C_NDB, XWBSB_TYPE(R5) ; Setup structure type
90 90 0C27 2720 MOVB #XWBSB_STA_CLO, XWBSB_STA(R5) ; Init logical-link state
80 80 0C2B 2721 MOVB #XWBSM_STS_CON, XWBSM_STS(R5) ; Init the status word
80 80 0C2F 2722 MOVB #XWBSM_FLG_CLO, XWBSM_FLG(R5) ; Init FLG bits
1C A5 0200 8F 80 0C35 2723 MOVB #NETSC_DR_INVALID,XWBSW_R_REASON(R5) ; Init rcv'd discon reason
44 A5 0064 8F 80 0C3B 2724 MOVB #NETSC_DR_INVALID,XWBSW_X_REASON(R5) ; Init xmt'd discon reason
46 A5 0064 8F 80 0C41 2725 MOVB R4, XWBSW_REMNOD(R5) ; Setup remote node i.d.
3A A5 54 80 0C45 2726 MOVL R2, XWBSL_VCB(R5) ; Setup VCB address
30 A5 52 D0 0C49 2727 MOVB #*X<80>, XWBS$+ACBSB_RMOD(R5) ; Setup Special Kernal AST
0178 C5 00000000'EF 9E 0C4F 2728 MOVAB NET$KAST, XWBS$+ACBSL_KAST(R5) ; mode and address
0C58 2729
50 0118 C5 9E 0C58 2730 MOVAB XWBSQ_FREE_CXB(R5),R0 ; Get free queue address
60 50 D0 0C5D 2731 MOVL R0,(R0) ; Init queue header
60 80 DE 0C60 2732 MOVAL (R0)+,(R0)
0C63 2733
54 A5 63 A2 9B 0C63 2734 MOVZBW RCB$B_ECL_RFA(R2), XWBSW_RETRAN(R5) ; Set default rexmit's
56 A5 64 A2 9B 0C68 2735 MOVZBW RCB$B_ECL_DFA(R2), XWBSW_DLY_FACT(R5) ; Set default delay factor
58 A5 65 A2 9B 0C6D 2736 MOVZBW RCB$B_ECL_DWE(R2), XWBSW_DLY_WGHT(R5) ; Set default delay weight
42 A5 7C A2 80 0C72 2737 MOVW RCB$W_ECLSEGSIZ(R2),XWBSW_REMSIZ(R5) ; Set temp 'seg' size
50 A5 76 A2 01 A1 0C77 2738 ADDW3 #1,RCB$W_TIM_CNI(R2),XWBSW_TIMER(R5) ; Set inbound connect timer
0C7D 2739 ; (#1 is for clock skew)
0C7D 2740
0C7D 2741
0C7D 2742
0C7D 2743
0C7D 2744
51 015E C5 9E 0C7D 2745 MOVAB XWBSI_TR3HDR+6(R5),R1 ; Setup route-header pointer
71 71 94 0C82 2746 CLRB -(R1) ; Zero the 'visits' field
71 0E A2 80 0C84 2747 MOVW RCB$W_ADDR(R2),-(R1) ; Enter src node address
71 54 80 0C88 2748 MOVW R4, -(R1) ; Enter dst node address
71 02 90 0C8B 2749 MOVB #TR3$C_MSG_DATA, -(R1) ; Enter message type
0120 C5 51 D0 0C8E 2750 MOVL R1,XWBSL_PTR_RTHD(R5) ; Setup route-header pointer
71 06 D0 0C93 2751 MOVL #6, -(R1) ; Store the route-header size
05 05 0C96 2752 RSB ; Done
0C97 2753
```

```

OC97 2755 .SBTTL XWB_LOCLNK - Get XWB via local link number
OC97 2756 :++
OC97 2757 :
OC97 2758 : INPUTS: R5 Any NET UCB address
OC97 2759 : R3 Local link number
OC97 2760 :
OC97 2761 : OUTPUTS: R5 Address of associated XWB, or low bit set if none
OC97 2762 :
OC97 2763 : All other registers are preserved.
OC97 2764 :--
OC97 2765 XWB_LOCLNK:
14 BB OC97 2766 PUSHR #^M<R2,R4> ; Get XWB context
OC99 2767 ; Save reg
52 34 A5 D0 OC99 2768 MOVL UCBSL_VCB(R5),R2 ; Get RCB address
OC9D 2769 BNEQ 5$ ; If NEQ the RCB exists
55 01 D0 OC9F 2770 MOVL #1,R5 ; Invalidate XWB address
OC9A 2771 BRB 10$ ; Done
OC9A 2772 5$: BSBB NET$XWB_LOCLNK ; If NEQ Locate the link
OC9A 2773
OC9A 2774 14 BA OCA6 2774 10$: POPR #^M<R2,R4> ; Restcre reg
OC9A 2775 05 OCA8 2775 RSB
OC9A 2776
OC9A 2777
OC9A 2778 .SBTTL NET$XWB_LOCLNK - Get XWB via local link number
OC9A 2779 :++
OC9A 2780 :
OC9A 2781 : The Link Table is located and the slot associated with the specified link
OC9A 2782 : number is found. If this slot contains an XWB then the link sequence number
OC9A 2783 : is checked. If there is a sequence number mismatch, or if there is no
OC9A 2784 : active XWB, then the low bit of R5 is set. Else, the XWB address is stored
OC9A 2785 : in R5.
OC9A 2786 :
OC9A 2787 :
OC9A 2788 : INPUTS: R5,R4 Scratch
OC9A 2789 : R3 Local link number - high order word is clear
OC9A 2790 : R2 RCB address
OC9A 2791 :
OC9A 2792 : OUTPUTS: R5 Address of associated XWB, or low bit set if none
OC9A 2793 : R4 LTB (link table) address
OC9A 2794 :
OC9A 2795 : All other registers are preserved.
OC9A 2796 :
OC9A 2797 :--
OC9A 2798 NET$XWB_LOCLNK::
54 24 A2 D0 OC9A 2799 MOVL RCB$SL_PTR_LTB(R2),R4 ; Locate XWB via local link number
OC9A 2800 BEQL 20$ ; Get Link Table pointer
55 53 FFFFFFFC00 8F CB OCAF 2801 BICL3 #^C<NET$C_MAXLNK>,R3,R5 ; Return error if not there
OC9A 2802 BEQL 20$ ; Get link 'index'
OC9A 2803 CMPW R5,LTB$W_SLT_TOT(R4) ; Index '0' isn't used
OC9A 2804 BGTRU 20$ ; Index within range ?
55 10 A4 45 D0 OCBF 2805 MOVL LTB$SL_SLOTS(R4)[R5],R5 ; If not, branch
OC9A 2806 BLBS R5,30$ ; Get XWB address
OC9A 2807 CMPW R3,XWB$W_LOCLNK(R5) ; If LBS then none
OC9A 2808 BEQL 30$ ; Sequence number match ?
OC9A 2809 BISB #1,R5 ; If so, branch
OC9A 2810 30$: RSB ; Flag no associated XWB
OC9A 2810 30$: RSB
```

```

OCD1 2812 .SBTTL NET$RET_SLOT - Return logical-link XWB slot if done
OCD1 2813 .SBTTL NET$QUE_XWB - Queue XWB to NETACP's AQB
OCD1 2814 :++
OCD1 2815
OCD1 2816 : If the XWB is busy then the queue attempt is aborted. If the XWB is
OCD1 2817 : not busy then the XWB$V_STS_SOL bit is set to prevent any further XWB use.
OCD1 2818 :
OCD1 2819 :
OCD1 2820 :
OCD1 2821 : INPUTS: R5 XWB pointer
OCD1 2822 :
OCD1 2823 : OUTPUTS: R0,R1 Zero
OCD1 2824 :
OCD1 2825 : All other registers are preserved.
OCD1 2826 :
OCD1 2827 :
OCD1 2828 :--
OCD1 2829 NET$RET_SLOT::
1E A5 00 91 OCD1 2830 CMPB #XWB$C_STA_CLO,XWB$B_STA(R5) : Return logical-link if done
1E A5 06 13 OCD5 2831 BEQL 10$ : In 'closed' state?
OC A5 06 91 OCD7 2832 CMPB #XWB$C_STA_DIR,XWB$B_STA(R5) : If so, continue
OC A5 11 12 OCDB 2833 : If DIR state then we've sent
OC A5 B5 12 OCDD 2834 BNEQ 40$ : the DC msg already
OC A5 12 OCE0 2835 10$: TSTW XWB$W_REFCNT(R5) : If not, XWB is still active
OE A5 OC04 8F B3 OCE2 2836 BNEQ 40$ : Any references?
OE A5 OC04 8F B3 OCE2 2837 :& BBS #XWB$V_FLG_LOCK,XWB$W_FLG(R5),40$ : If NEQ must wait
OE A5 OC04 8F B3 OCE2 2838 BITW #XWB$M_STS_ASTPND!- : Exit if XWB is locked
OE A5 OC04 8F B3 OCE2 2839 : AST pending
OE A5 OC04 8F B3 OCE2 2840 : AST requested
OE A5 OC04 8F B3 OCE2 2841 BNEQ 40$ : Fork block in use
OE A5 OC04 8F B3 OCE2 2842 BSBB NET$DRAIN_FREE_CXB : If NEQ, XWB is busy
OE A5 OC04 8F B3 OCE2 2843 BSBB NET$QUE_XWB : Drain CXB free queue
OE A5 OC04 8F B3 OCE2 2844 40$: CLRQ R0 : Queue XWB to NETACP's AQB
OE A5 OC04 8F B3 OCE2 2845 RSB : Say 'nothing to xmit'
OE A5 OC04 8F B3 OCE2 2846 : Done
OE A5 OC04 8F B3 OCE2 2847
OE A5 OC04 8F B3 OCF1 2848 NET$QUE_XWB:: : Queue XWB to NETACP's AQB
OE A5 OC04 8F B3 OCF1 2849 ASSUME IPL$_SYNCH EQ NET$C_IPL
OE A5 OC04 8F B3 OCF1 2850
OE A5 OC04 8F B3 OCF1 2851 BBSS #XWB$V_STS_SOL,XWB$W_STS(R5),50$ : If BS, then queue block in use
OE A5 OC04 8F B3 OCF6 2852 PUSHF #^M<R2,R3,R4,R5> : Save regs
OE A5 OC04 8F B3 OCF8 2853
OE A5 OC04 8F B3 OCF8 2854 MOVL XWB$L_VCB(R5),R2 : Get RCB
OE A5 OC04 8F B3 OCF8 2855 INCW RCB$W_TRANS(R2) : Account for ACP transaction
OE A5 OC04 8F B3 OCF8 2856 MOVL RCB$L_AQB(R2),R4 : Get AQB
OE A5 OC04 8F B3 OCF8 2857 INSQUE (R5),AQB$L_ACPQBL(R4) : Queue XWB to AQB
OE A5 OC04 8F B3 OCF8 2858 BNEQ 30$ : If NEQ then not first
OE A5 OC04 8F B3 OCF8 2859 MOVL AQB$L_ACPPID(R4),R1 : Get ACP's PID
OE A5 OC04 8F B3 OCF8 2860 JSB G^SCH$WAKE : Wake the ACP
OE A5 OC04 8F B3 OCF8 2861
OE A5 OC04 8F B3 OCF8 2862 30$: POPR #^M<R2,R3,R4,R5> : Restore regs
OE A5 OC04 8F B3 OCF8 2863 50$: RSB : done
OE A5 OC04 8F B3 OCF8 2864
OE A5 OC04 8F B3 OCF8 2865
OE A5 OC04 8F B3 OCF8 2866
OE A5 OC04 8F B3 OCF8 2867 .SBTTL NET$DRAIN_FREE_CXB - Drain CXB free queue
OE A5 OC04 8F B3 OCF8 2868
```

```

      OD16 2869 NET$DRAIN_FREE_CXB:: ; Drain CXB free queue
      OD16 2870 :
      OD16 2871 :
      OD16 2872 : All registers except for R0 must be preserved.
      OD16 2873 :
      OD16 2874 :
50 0118 D5 0F OD16 2875 10$: REMQUE @XWBSQ_FREE_CXB(R5),R0 ; Get next CXB
      05 1D OD18 2876 BVS 20$ ; If VS, none left
      002E 30 OD1D 2877 BSBW NET$DEALLOCATE ; Deallocate block in R0
      F4 11 OD20 2878 BRB 10$ ; Loop
      05 05 OD22 2879 20$: RSB ; Done
      OD23 2880
      OD23 2881

```

```
0D23 2883 .SBTTL NET$ALONPGD_Z - Allocate and zero from system pool
0D23 2884 .SBTTL NET$ALONONPAGED - Allocate from system pool
0D23 2885 :++
0D23 2886 :
0D23 2887 : A buffer is allocated from non-paged pool and its size field is set to
0D23 2888 : the size requested. Its type field is set to DYN$C_CXB.
0D23 2889 :
0D23 2890 :
0D23 2891 : INPUTS: R2 = Scratch
0D23 2892 :          R1 = Size, in bytes, of block to be allocated
0D23 2893 :          R0 = Scratch
0D23 2894 :
0D23 2895 : OUTPUTS: R2 = Address of block if successful
0D23 2896 :           Zero if unsuccessful
0D23 2897 :           R0 = Standard VMS status code
0D23 2898 :
0D23 2899 : All other registers are preserved.
0D23 2900 :
0D23 2901 :--
0D23 2902 .ENABL LSB
0D23 2903 NET$ALONPGD_Z::
0D23 2904 BSBB NET$ALONONPAGED ; Allocate and zero non-paged buffer
25 50 E9 0D25 2905 BLBC R0,20$ ; Allocate the buffer
0D28 2906 ; If LBC then error
0D28 2907 PUSHR #^M<R0,R1,R2,R3,R4,R5> ; Save regs
0D2A 2908 MOVCS #0,(SP),#0,R1,(R2) ; Zero the entire buffer
0D30 2909 POPR #^M<R0,R1,R2,R3,R4,R5> ; Restore regs
0D32 2910 ;
11 11 0D32 2911 BRB 10$ ; Setup the type and size fields (again)
0D34 2912 ;
0D34 2913 ;
0D34 2914 NET$ALONONPAGED:: ; Allocate non-paged memory
0D34 2915 ;
0A BB 0D34 2916 PUSHR #^M<R1,R3> ; Save regs
00000000 GF 16 0D36 2917 JSB G^EXE$ALONONPAGED ; Allocate memory
0A BA 0D3C 2918 POPR #^M<R1,R3> ; Restore regs
0D3E 2919 ;
04 50 E8 0D3E 2920 BLBS R0,0$ ; If LBS then success
52 D4 0D41 2921 CLRL R2 ; Zero the buffer pointer
08 11 0D43 2922 BRB 20$ ; Take common exit
08 A2 51 B0 0D45 2923 10$: MOVW R1,CXB$W SIZE(R2) ; Set size for deallocation
1B 90 0D49 2924 MOVB #DYN$C_CXB,- ;
0A A2 0D4B 2925 CXB$B_TYPE(R2) ; Set tentative buffer type
05 0D4D 2926 20$: RSB ; Return with status in R0
0D4E 2927 ;
0D4E 2928 .DSABL LSB
0D4E 2929
```

```
OD4E 2931 .SBTTL NET$DEALLOCATE - Deallocate non-paged pool
OD4E 2932 :+
OD4E 2933 :
OD4E 2934 : IPL must be NET$_IPL or lower.
OD4E 2935 :
OD4E 2936 :
OD4E 2937 : INPUTS:      R0      Address of block
OD4E 2938 :
OD4E 2939 : OUTPUTS:    R0      Zero
OD4E 2940 :
OD4E 2941 : All other registers are preserved.
OD4E 2942 :
OD4E 2943 :-
OD4E 2944 : ASSUME NET$_IPL LE IPL$_SYNCH ; Can't deallocate above SYNCH
OD4E 2945 :
OD4E 2946 NET$DEALLOCATE:: : Deallocate non-paged pool
OE BB OD4E 2947 PUSHF #M<R1,R2,R3> : Save regs
7E D4 OD50 2948 CLRL -(SP) : Value to return in R0
OD52 2949 :
51 0B A0 3C OD52 2950 MOVZWL 8(R0),R1 : Get size of block
00000000'GF 16 OD56 2951 JSB G^EXE$DEANONPGDSIZ : Deallocate it
OD5C 2952 :
OF BA OD5C 2953 POPR #M<R0,R1,R2,R3> : Restore regs
05 OD5E 2954 RSB : Done
OD5F 2955 :
```

```

OD5F 2957 .SBTTL NET$MOV_TO_XWB - Move counted string to XW$B_DATA
OD5F 2958 .SBTTL NET$MOV_CSTR - Move counted string with count field
OD5F 2959 .SBTTL NET$MOV_USTR - Move counted string without count field
OD5F 2960 :+
OD5F 2961 :
OD5F 2962 : The source string is moved to its destination. Both the source
OD5F 2963 : and destination pointers are updated.
OD5F 2964 :
OD5F 2965 :
OD5F 2966 : INPUTS: R5 Pointer to destination field
OD5F 2967 : R1 Pointer to count field of source string
OD5F 2968 :
OD5F 2969 : OUTPUTS: R3 Pointer to first byte beyond end of destination
OD5F 2970 : R1 Pointer to first byte beyond source string
OD5F 2971 :
OD5F 2972 : All other registers are preserved
OD5F 2973 :
OD5F 2974 :
OD5F 2975 :-
OD5F 2976 .ENABL LSB
OD5F 2977 NET$MOV_TO_XWB:: : Move counted string to XW$B_DATA
53 53 DD OD5F 2978 PUSH R3 : Save reg
53 5B A5 9E OD61 2979 MOVAB XW$B_DATA(R5),R3 : Setup destination ptr
04 10 OD65 2980 BSBB NET$MOV_CSTR : Move the string
53 8E D0 OD67 2981 POPL R3 : Restore reg
05 OD6A 2982 RSB : Done
OD6B 2983 :
OD6B 2984 NET$MOV_CSTR:: : Move counted string with count byte
35 BB OD6B 2985 PUSH R4,R5 : Save regs
OD6D 2986 :
50 61 9B OD6D 2987 MOVZBW (R1),R0 : Get string length
50 B6 OD70 2988 INCW R0 : Include count itself
05 11 OD72 2989 BRB 10$ : Continue in common
OD74 2990 :
OD74 2991 NET$MOV_USTR:: : Mov counted str w/o count byte
35 BB OD74 2992 PUSH R4,R5 : Save regs
OD76 2993 :
63 50 81 9B OD76 2994 MOVZBW (R1)+,R0 : Get count value, advance ptr
63 61 50 28 OD79 2995 10$: MOV C3 R0,(R1),(R3) : Move the string
OD7D 2996 :
35 BA OD7D 2997 POP R4,R5 : Restore regs
05 OD7F 2998 RSB
OD80 2999 :
OD80 3000 .DSABL LSB
OD80 3001
```

```

0D80 3003 .SBTTL NET$POST_IO - Send IRP to COM$POST
0D80 3004 :+
0D80 3005 :
0D80 3006 : INPUTS: R3 IRP address
0D80 3007 : R0 Scratch
0D80 3008 :
0D80 3009 : OUTPUTS: R0 SSS_NORMAL
0D80 3010 :
0D80 3011 : All other registers are preserved
0D80 3012 :-
0D80 3013 NET$POST_IO:: ; Send IRP to COM$POST
0D80 3014 :
0D80 3015 :
0D80 3016 : Complete the I/O
0D80 3017 :
0D80 3018 :
0D80 3019 PUSHL R5 ; Save XWB pointer
0D82 3020 MOVL IRP$L_UCB(R3),R5 ; Get UCB address
0D86 3021 JSB G^COM$POST ; Another packet for the heap
0D8C 3022 MOVL S^#SSS_NORMAL,R0 ; Always return success
0D8F 3023 POPL R5 ; Recover XWB address
0D92 3024 RSB ; Done
0D93 3025
0D93 3026
0D93 3027
0D93 3028
0D93 3029 .END

```

55 1C 55 DD
00000000 GF 16
50 01 D0
55 8ED0
05

NETDRVSES
Symbol table

F 1

- DECnet Session Control Module for NETD 16-SEP-1984 01:32:10 VAX/VMS Macro V04-00 Page 70
5-SEP-1984 02:20:26 [NETACP.SRC]NETDRVSES.MAR;1 (59)

\$\$\$	= 00000020	R	02	ACT\$RCV_DATA	= 00000009		
\$\$OP	= 00000002			ACT\$RCV_DTACK	= 0000000A		
\$\$NSPMMSG	= 00000000			ACT\$RCV_DX	= 0000000D		
\$\$TR3MSG	= 00000000			ACT\$RCV_LI	= 0000000B		
\$\$TR4MSG	= 00000000			ACT\$RCV_LIACK	= 0000000C		
ABORT	000009AB	R	03	ACT\$RCV_RTS	= 00000008		
ABORT_ALL	00000A08	R	03	ACT\$RES_DISC	= 00000010		
ACBSB_RMOD	= 0000000B			ACT\$RTS_NLT	= 00000006		
ACBSC_LENGTH	= 0000001C			ACT\$SHR[ENK	= 00000014		
ACBSL_KAST	= 00000018			ACT\$SSABORT	= 00000012		
ACBSL_PID	= 0000000C			ACT DISPATCH	00000447	R	03
ACPSACCESSNET	*****	X	03	AQBSL_ACPPIID	= 0000000C		
ACPSC_STA_F	= 00000004			AQBSL_ACPQBL	= 00000004		
ACPSC_STA_H	= 00000005			AT\$NULL	*****	X	02
ACPSC_STA_I	= 00000000			BIT:::	= 00000004		
ACPSC_STA_N	= 00000001			BRDCST	00000A53	R	03
ACPSC_STA_R	= 00000002			BUG\$NETNOSTATE	*****	X	03
ACPSC_STA_S	= 00000003			CHANGE STA	0000036F	R	03
ACPSDEACCESS	*****	X	03	CHKRETAADDR	00000345	R	03
ACPSMODIFY	*****	X	03	CHK X IRP	000007D1	R	03
ACT\$ABORT	*****	X	03	CLEANOP ACCESS	00000814	R	03
ACT\$BUG	0000047A	R	03	CNFS_ADVANCE	= 00000000		
ACT\$CANLNK	*****	X	03	CNFS_QUIT	= 00000002		
ACT\$CONFIRM	00000618	RG	03	CNFS_TAKE_CURR	= 00000003		
ACT\$DEACCESS	0000078C	RG	03	CNFS_TAKE_PREV	= 00000001		
ACT\$ENT_RUN	0000070B	RG	03	COM\$POST	*****	X	03
ACT\$INITIATE	0000064F	RG	03	CRBSL_INTD	= 00000024		
ACT\$LOG	0000047E	R	03	CRE_LNK	000009DF	R	03
ACT\$NOLINK	00000484	R	03	CXBSB_R_AREA	00000039		
ACT\$NOP	00000479	R	03	CXBSB_R_FLG	00000038		
ACT\$RCV_CA	*****	X	03	CXBSB_R_NSPTYP	00000039		
ACT\$RCV_CC	*****	X	03	CXBSB_TYPE	= 0000000A		
ACT\$RCV_CI	*****	X	03	CXBSB_X_NSPTYP	= 0000004E		
ACT\$RCV_CR	*****	X	03	CXBSB_DCL	= 00000020		
ACT\$RCV_DATA	*****	X	03	CXBSB_HEADER	= 00000048		
ACT\$RCV_DTACK	*****	X	03	CXBSB_R_LENGTH	= 0000003C		
ACT\$RCV_DX	*****	X	03	CXBSL_LINK	= 00000010		
ACT\$RCV_LI	*****	X	03	CXBSL_R_MSG	0000002C		
ACT\$RCV_LIACK	*****	X	03	CXBSL_R_RCB	00000028		
ACT\$RCV_RTS	*****	X	03	CXBSL_DCL	= 00000028		
ACT\$RES_DISC	000007B2	RG	03	CXBSL_X_DATA	00000057		
ACT\$RTS_NLT	*****	X	03	CXBSL_X_XPORT	00000048		
ACT\$SHR[ENK	0000048B	R	03	CXBSW_R_ADJ	0000003A		
ACT\$SSABORT	0000048B	R	03	CXBSW_R_BCNT	00000030		
ACT\$ABORT	= 0000000E			CXBSW_R_DSTNOD	00000034		
ACT\$BUG	= 00000000	G		CXBSW_R_NSPSEQ	0000003A		
ACT\$CANLNK	= 0000000F			CXBSW_R_PATH	00000032		
ACT\$CONFIRM	= 00000013			CXBSW_R_SRCNOD	00000036		
ACT\$DEACCESS	= 00000015			CXBSW_SIZE	= 00000008		
ACT\$ENT_RUN	= 00000007			CXBSW_X_NSPACK	00000053		
ACT\$INITIATE	= 00000011			CXBSW_X_NSPLOC	00000051		
ACT\$LOG	= 00000000			CXBSW_X_NSPREM	0000004F		
ACT\$NOP	= 00000004			CXBSW_X_NSPSEQ	00000055		
ACT\$RCV_CA	= 00000003			DDBSL_DDT	= 0000000C		
ACT\$RCV_CC	= 00000005			DDBST_NAME	= 00000014		
ACT\$RCV_CI	= 00000002			DEAL_ICB	00000861	R	03
ACT\$RCV_CR	= 00000001			DECLARE	00000AA5	R	03

NETDRVSES
Symbol table

G 1

- DECnet Session Control Module for NETD 16-SEP-1984 01:32:10 VAX/VMS Macro V04-00
5-SEP-1984 02:20:26 [NETACP.SRC]NETDRVSES.MAR;1

Page 71
(59)

NE1
V04

DEVSM_AVL	*****	X	02	IOS_ACPCONTROL	=	00000038	
DEVSM_IDV	*****	X	02	IOS_DEACCESS	=	00000034	
DEVSM_MBX	*****	X	02	IOS_READBLK	=	00000021	
DEVSM_NET	*****	X	02	IOS_READVBLK	=	00000031	
DEVSM_ODV	*****	X	02	IOS_SETMODE	=	00000023	
DISC ONE	000009FD	R	03	IOS_VIRTUAL	=	0000003F	
DLLTRN	00000AD8	R	03	IOS_WRITEBLK	=	00000020	
DPTSB_FLAGS	= 0000000D			IOS_WRITEVBLK	=	00000030	
DPTSC_LENGTH	= 00000038			IOCSINITIATE	*****	X	03
DPTSC_VERSION	= 00000004			IOCSMNTVER	*****	X	03
DPTSMINITAB	00000038	R	02	IOCSREQCOM	*****	X	03
DPTSM_NOUNLOAD	= 00000004			IOCSRETURN	*****	X	03
DPTSMREINITAB	00000074	R	02	IPLS_SYNCH	=	00000008	
DPTSTAB	00000000	R	02	IPLS_TIMER	=	00000008	
DRAIN_RCV	00000921	R	03	IRPSC_BCNT	=	00000032	
DRAIN_XMT	000008C2	R	03	IRPSL_DIAGBUF	=	0000004C	
DYNSC_CRB	= 00000005			IRPSL_IOQFL	=	00000000	
DYNSC_CXB	= 0000001B			IRPSL_IOST1	=	00000038	
DYNSC_DDB	= 00000006			IRPSL_PID	=	0000000C	
DYNSC_DPT	= 0000001E			IRPSL_SVAPTE	=	0000002C	
DYNSC_NDB	= 0000001C			IRPSL_UCB	=	0000001C	
DYNSC_ORB	= 00000049			IRPSL_WIND	=	00000018	
DYNSC_UCB	= 00000010			IRPSM_FUNC	=	00000002	
EXESABORTIO	*****	X	03	IRPSQ_NT PRVMSK	=	00000040	
EXESALONONPAGED	*****	X	03	IRPSV_COMPLX	=	00000003	
EXESDEANONPGDSIZ	*****	X	03	IRPSW_BCNT	=	00000032	
EXESFINISHIO	*****	X	03	IRPSW_CHAN	=	00000028	
EXESFORK	*****	X	03	IRPSW_FUNC	=	00000020	
EXESGO SYSTIME	*****	X	03	IRPSW_STS	=	0000002A	
EXESINSTIMO	*****	X	03	JIBSL_BYTCNT	=	00000020	
EXESWRTMAILBOX	*****	X	03	JIBSL_BYTLM	=	00000024	
EXIT	000009BA	R	03	JIBSW_FILCNT	=	00000030	
FKBSC_LENGTH	= 00000018			LPDSB_PTH_INX	=	00000020	
FUNCTABLE	00000038	R	03	LPDSC_LOC_INX	=	00000001	
FUNCTAB_LEN	= 00000058			LSB	=	00000000	
GET_P1DSC	000007F2	R	03	LSBSB_R_CXBCNT	=	00000028	
GET_P2DSC	000007F7	R	03	LSBSB_R_CXBQUO	=	00000029	
GET_P3DSC	000007FC	R	03	LSBSB_SPARE	=	0000002A	
GET_P4DSC	00000801	R	03	LSBSB_STS	=	0000002B	
GET_WNDSC	000007EE	R	03	LSBSB_X_ADJ	=	0000000B	
ICBSB_DATA	= 0000007C			LSBSB_X_CXBACT	=	0000000D	
ICBSB_RID	= 00000092			LSBSB_X_CXBCNT	=	0000000F	
ICBSC_RID	= 00000010			LSBSB_X_CXBQUO	=	0000000E	
ICBST_RID	= 00000093			LSBSB_X_PKTWND	=	0000000C	
ICBSW_DLY_FACT	= 0000000E			LSBSB_X_REQ	=	0000000A	
ICBSW_DLY_WGHT	= 00000010			LSBSL_CROSS	=	0000002C	
ICBSW_LOCLNK	= 00000002			LSBSL_R_CXB	=	00000020	
ICBSW_PATH	= 00000000			LSBSL_R_IRP	=	0000001C	
ICBSW_RETRAN	= 0000000C			LSBSL_X_CXB	=	00000018	
ICBSW_SEGSIZ	= 00000012			LSBSL_X_IRP	=	00000014	
ICBSW_TIM_INACT	= 00000006			LSBSL_X_PND	=	00000010	
ICBSW_TIM_OCON	= 00000004			LSBSM_BOM	=	00000020	
INIT_XWB	00000C1F	R	03	LSBSM_EOM	=	00000040	
IOSM_FCODE	= 0000003F			LSBSM_LI	=	00000001	
IOSM_INTERRUPT	= 00000040			LSBSS_LSB	=	00000030	
IOSV_ABORT	= 00000008			LSBSS_SPARE	=	00000004	
IOS_ACCESS	= 00000032			LSBSS_STS	=	00000001	

NETDRVSES
Symbol table

H 1
- DECnet Session Control Module for NETD 16-SEP-1984 01:32:10 VAX/VMS Macro V04-00 Page 72
5-SEP-1984 02:20:26 [NETACP.SRC]NETDRVSES.MAR;1 (59)

```
LSBSV_BOM      = 00000005
LSBSV_EOM      = 00000006
LSBSV_LI       = 00000000
LSBSV_SPARE    = 00000001
LSBSW_HAA      = 00000008
LSBSW_HAR      = 00000006
LSBSW_HAX      = 00000026
LSBSW_HNR      = 00000024
LSBSW_HXS      = 00000004
LSBSW_LNX      = 00000002
LSBSW_LUX      = 00000000
LTBSL_SLOTS    = 00000010
LTBSL_XWB      = 0000000C
LTBSW_SLT_TOT  = 00000004
MASKH          = 01000000
MASKL          = 00000000
MBXSM_EVTAVL   = 00000002
MBXSM_EVTRCVCHG = 00000004
MBXSM_EVTXMTCHG = 00000008
MBXSM_NETSTATE = 00000001
MBXSV_EVTAVL   = 00000001
MBXSV_EVTRCVCHG = 00000002
MBXSV_EVTXMTCHG = 00000003
MBXSV_NETSTATE = 00000000
MBX_TABLE      = 000001E8 R      03
MSG$_ABORT     = 00000030
MSG$_CONNECT   = 00000032
MSG$_DISCON    = 00000033
MSG$_EVTAVL    = 0000003E
MSG$_EVTRCVCHG = 0000003F
MSG$_EVTXMTCHG = 00000044
MSG$_EXIT      = 00000034
MSG$_NETSHUT   = 0000003B
MSG$_PATHLOST  = 00000036
MSG$_REJECT    = 00000038
MSG$_THIRDPARTY = 00000039
NET$AB_STTAB   = 0000013C R      03
NET$ACCESS     = 000005B1 R      03
NET$ACK_XMT_SEGS = ***** X      03
NET$ACP_COMM    = 00000969 RG     03
NET$ALONONPAGED = 00000D34 RG     03
NET$ALONPGD_2   = 00000D23 RG     03
NET$ALTENTRY    = ***** X      03
NET$AW_FLG_CLRM = 0000012C R      03
NET$AW_FLG_SETM = 0000011C R      03
NET$AZ_DR_CONTAB = 00000262 R      03
NET$AZ_DR_TABLE = 00000204 R      03
NET$CANCE[     = 0000087B R      03
NET$CHK_X_IDLE  = 000007C2 RG     03
NET$CMP[ ACC    = 000006CA RG     03
NET$COMPLEX_EV  = 00000330 RG     03
NET$CONTROL     = 0000054A R      03
NET$CREATE_XWB  = 00000BC6 RG     03
NET$CTLR_INIT   = 000002E1 R      03
NET$C_ACTBITS   = 00000005
NET$C_ACT_TIMER = 0000001E
NET$C_DR_ABORT  = 00000009
```

```
NET$C_DR_ACCESS = 00000022
NET$C_DR_BUSY   = 00000006
NET$C_DR_DEACC  = 00000066 G
NET$C_DR_EXIT   = 00000026
NET$C_DR_FMT    = 00000005
NET$C_DR_INVALID = 00000064 G
NET$C_DR_IVNODE = 00000002
NET$C_DR_NOBJ   = 00000004
NET$C_DR_NONODE = 00000002
NET$C_DR_NOPATH = 00000027
NET$C_DR_NORMAL = 00000000
NET$C_DR_RSU    = 00000001
NET$C_DR_SHUT   = 00000003
NET$C_DR_THIRD  = 00000008
NET$C_EFN_ASYN  = 00000002
NET$C_EFN_WAIT  = 00000001
NET$C_IPL       = 00000008
NET$C_MAXACCFD  = 00000027
NET$C_MAXLINNAM = 0000000F
NET$C_MAXLNK    = 000003FF
NET$C_MAXNODNAM = 00000006
NET$C_MAXOBJNAM = 0000000C
NET$C_MAX_AREAS = 0000003F
NET$C_MAX_LINES = 00000040
NET$C_MAX_NCB   = 0000006E
NET$C_MAX_NODES = 000003FF
NET$C_MAX_OBJ   = 000000FF
NET$C_MAX_WQE   = 00000014
NET$C_MINBUFSIZ = 000000C0
NET$C_STABITS   = 00000003
NET$C_TID_ACT   = 00000003
NET$C_TID_RUS   = 00000001
NET$C_TID_XRT   = 00000002
NET$C_TRCTL_CEL = 00000002
NET$C_TRCTL_OVR = 00000005
NET$C_UTLBUFSIZ = 00001000
NET$DDT         = 00000000 RG     03
NET$DEACCESS    = 00000777 RG     03
NET$DEALLOCATE  = 00000D4E RG     03
NET$DRAIN_FREE_CXB = 00000D16 RG     03
NET$SEND        = ***** X      02
NET$SEND_EVENT  = 0000032A RG     03
NET$SEVENT      = 00000356 RG     03
NET$FDT_ACCESS  = 000005A3 R      03
NET$FDT_CONTROL = 00000538 R      03
NET$FDT_DEACCESS = 00000711 RG     03
NET$FDT_RCV     = ***** X      03
NET$FDT_SETMODE = 0000050B R      03
NET$FDT_XMT     = ***** X      03
NET$FORK        = 000002EC RG     03
NET$GL_OFF_DPTFLG = 00000118 RG     03
NET$GL_WORKBITS = 000001E4 RG     03
NET$GQ_PATCH    = 00000090 RG     03
NET$INTERRUPT   = 000002E1 R      03
NET$KAST        = ***** X      03
NET$MAP_R_REASON = 000002C0 RG     03
NET$MARK_LINK   = ***** X      03
```

NETDRVSES
Symbol table

1 1

- DECnet Session Control Module for NETD 16-SEP-1984 01:32:10 VAX/VMS Macro V04-00 Page 73
5-SEP-1984 02:20:26 [NETACP.SRC]NETDRVSES.MAR;1 (59)

```

NET$MOV_CSTR      00000D6B RG 03
NET$MOV_TO_XWB    00000D5F RG 03
NET$MOV_USTR      00000D74 RG 03
NET$M_MAXLNKMSK  = 000003FF
NET$M_STAMSK      = 000000E0
NET$POST_IO       00000D80 RG 03
NET$PRE_EMPTY     00000341 RG 03
NET$PURG_RUN      000008AF RG 03
NET$QUE_XWB       00000CF1 RG 03
NET$RCV_DONE      ***** X 03
NET$RESET_TIMER   ***** X 03
NET$RET_SCOT      00000CD1 RG 03
NET$SCH_MSG       000003A3 RG 03
NET$SEND_CS_MBX   00000B31 RG 03
NET$SEND_MBX      00000B71 RG 03
NET$SETUP_RUN     ***** X 03
NET$STARTIO       00000490 R 03
NET$TIMER         ***** X 03
NET$UNIT_INIT     000002E2 R 03
NET$UNSOC_INTR    ***** X 03
NET$XMT_DONE      ***** X 03
NET$XWB_LOCLNK    00000CA9 RG 03
NETEVT$_CA        = 00000001 G
NETEVT$_CANLNK    = 0000000D G
NETEVT$_CC         = 00000002 G
NETEVT$_CCA        = 00000010 G
NETEVT$_CI         = 00000000 G
NETEVT$_CIA        = 0000000F G
NETEVT$_CRA        = 00000011 G
NETEVT$_DATA       = 00000005 G
NETEVT$_DC         = 0000000B G
NETEVT$_DEA        = 00000012 G
NETEVT$_DI         = 0000000A G
NETEVT$_DSCLNK     = 0000000C G
NETEVT$_DTACK      = 00000006 G
NETEVT$_INT        = 00000008 G
NETEVT$_LIACK      = 00000009 G
NETEVT$_LS         = 00000007 G
NETEVT$_MBXERR     = 00000013 G
NETEVT$_PH2CCS     = 00000003 G
NETEVT$_PROERR     = 00000014 G
NETEVT$_RESDIS     = 0000000E G
NETEVT$_RTS        = 00000004 G
NETUPD$_ABOLNK     = 00000008
NETUPD$_ABORT      = 00000001
NETUPD$_BPDCST     = 0000000A
NETUPD$_CONNECT    = 00000002
NETUPD$_CRELNK     = 00000007
NETUPD$_DLL_ON     = 00000005
NETUPD$_DSCLNK     = 00000009
NETUPD$_EXIT       = 00000003
NETUPD$_PROCRE     = 00000004
NETUPD$_REPLY      = 0000000B
NEW STATE          0000037E R 03
NSP$$$_QUAL_ACK    = 00000000
NSP$$$_QUAL_ALTFLW = 00000000
NSP$$$_QUAL_DATA   = 00000000

```

```

NSP$$$_QUAL_FLW    = 00000000
NSP$$$_QUAL_INF    = 00000000
NSP$$$_QUAL_MSG    = 00000000
NSP$$$_QUAL_SRV    = 00000000
NSP$C_EXT_LNK      = 0000001E
NSP$C_FLW_DATA     = 00000000
NSP$C_FLW_INT      = 00000001
NSP$C_FLW_NOP      = 00000000
NSP$C_FLW_XOFF     = 00000001
NSP$C_FLW_XON      = 00000002
NSP$C_HSZ_ACK      = 00000007
NSP$C_HSZ_CA       = 00000003
NSP$C_HSZ_CC       = 00000064
NSP$C_HSZ_CD       = 000000F0
NSP$C_HSZ_CI       = 000000F0
NSP$C_HSZ_DATA     = 00000009
NSP$C_HSZ_DC       = 00000016
NSP$C_HSZ_DI       = 00000016
NSP$C_HSZ_INT      = 00000009
NSP$C_HSZ_LS       = 00000009
NSP$C_INF_V31      = 00000001
NSP$C_INF_V32      = 00000000
NSP$C_INF_V33      = 00000002
NSP$C_MAXHDR       = 00000009
NSP$C_MAX_DELAY    = 00000014
NSP$C_MAX_R_CXB    = 00000007
NSP$C_MAX_XPW      = 00000007
NSP$C_MSG_CA       = 00000024
NSP$C_MSG_CC       = 00000028
NSP$C_MSG_CI       = 00000018
NSP$C_MSG_DATA     = 00000000
NSP$C_MSG_DC       = 00000048
NSP$C_MSG_DI       = 00000038
NSP$C_MSG_DTACK    = 00000004
NSP$C_MSG_INT      = 00000030
NSP$C_MSG_LIACK    = 00000014
NSP$C_MSG_LS       = 00000010
NSP$C_SRV_MFC      = 00000002
NSP$C_SRV_NFC      = 00000000
NSP$C_SRV_REQ      = 00000001
NSP$C_SRV_SFC      = 00000001
NSP$M_ACK_NAK      = 00001000
NSP$M_ACK_NUM      = 00000FFF
NSP$M_ACK_VALID    = 00008000
NSP$M_DATA_BOM     = 00000020
NSP$M_DATA_EOM     = 00000040
NSP$M_DATA_OVFW    = 00000080
NSP$M_FLW_CHAN     = 0000000C
NSP$M_FLW_DRV      = 000000F0
NSP$M_FLW_INT      = 00000020
NSP$M_FLW_INUSE    = 00000010
NSP$M_FLW_LISUB    = 00000004
NSP$M_FLW_MODE     = 00000003
NSP$M_FLW_SP1      = 00000008
NSP$M_FLW_SP2      = 00000040
NSP$M_FLW_SP3      = 00000080
NSP$M_FLW_XOFF     = 00000001

```

NETDRVSES
Symbol table

J 1
- DECnet Session Control Module for NETD 16-SEP-1984 01:32:10 VAX/VMS Macro V04-00 Page 74
5-SEP-1984 02:20:26 [NETACP.SRC]NETDRVSES.MAR;1 (59)

NSPSM_Flw_XON = 00000002
NSPSM_INF_VER = 00000003
NSPSM_MSG_INT = 00000020
NSPSM_MSG_LI = 00000010
NSPSM_SRV_01 = 00000003
NSPSM_SRV_EXT = 00000080
NSPSM_SRV_Flw = 0000000C
NSPSM_SRV_REQ = 000000F3
NSPSM_SRV_SP1 = 00000070
NSPSR_QUAL = 00000000
NSPSOLICIT *****
NSPSS_ACK_NUM = 0000000C
NSPSS_ACK_SP2 = 00000002
NSPSS_DATA_SP = 00000005
NSPSS_Flw_CHAN = 00000002
NSPSS_Flw_Drv = 00000004
NSPSS_Flw_Mode = 00000002
NSPSS_INF_VER = 00000002
NSPSS_MSG_SP1 = 00000004
NSPSS_NSMSG = 00000005
NSPSS_QUAL = 00000005
NSPSS_QUAL_ACK = 00000002
NSPSS_QUAL_ALTFLW = 00000001
NSPSS_QUAL_DATA = 00000001
NSPSS_QUAL_Flw = 00000001
NSPSS_QUAL_INF = 00000001
NSPSS_QUAL_MSG = 00000005
NSPSS_QUAL_SRV = 00000001
NSPSS_SRV_01 = 00000002
NSPSS_SRV_Flw = 00000002
NSPSS_SRV_SP1 = 00000003
NSPSV_ACK_NAK = 0000000C
NSPSV_ACK_NUM = 00000000
NSPSV_ACK_SP2 = 0000000D
NSPSV_ACK_VALID = 0000000F
NSPSV_DATA_BOM = 00000005
NSPSV_DATA_EOM = 00000006
NSPSV_DATA_OVFW = 00000007
NSPSV_DATA_SP = 00000000
NSPSV_Flw_CHAN = 00000002
NSPSV_Flw_Drv = 00000004
NSPSV_Flw_INT = 00000005
NSPSV_Flw_INUSE = 00000004
NSPSV_Flw_LISUB = 00000002
NSPSV_Flw_Mode = 00000000
NSPSV_Flw_SP1 = 00000003
NSPSV_Flw_SP2 = 00000006
NSPSV_Flw_SP3 = 00000007
NSPSV_Flw_XOFF = 00000000
NSPSV_Flw_XON = 00000001
NSPSV_INF_VER = 00000000
NSPSV_MSG_INT = 00000005
NSPSV_MSG_LI = 00000004
NSPSV_MSG_SP1 = 00000000
NSPSV_SRV_01 = 00000000
NSPSV_SRV_EXT = 00000007
NSPSV_SRV_Flw = 00000002

X 03

NSPSV_SRV_SP1 = 00000004
NSPSW_DSTLNK = 00000001
NSPSW_SRCLNK = 00000003
ORBSB_FLAGS = 0000000B
ORBSL_OWNER = 00000000
ORBSM_PROT_16 = 00000001
ORBSW_PROT = 00000018
P1 = 00000000
P2 = 00000004
P3 = 00000008
PATCH_AREA_SIZE = 00000080
PCBSL_JIB = 00000080
PCBSL_PHD = 0000006C
PCBSL_PID = 00000060
PHDSQ_PRIVMSK = 00000000
PR\$ IPL = 00000012
PROC RE = 0000099E
PROC IO = 000004DA
R3_OFF = 0000000C
R4_OFF = 00000010
R5_OFF = 00000014
RCBSB_ECL_DFA = 00000064
RCBSB_ECL_DWE = 00000065
RCBSB_ECL_RFA = 00000063
RCBSL_ACP_UCB = 00000014
RCBSL_AQB = 00000010
RCBSL_PTR_LTB = 00000024
RCBSL_PTR_TQE = 00000030
RCBSW_ADDR = 0000000E
RCBSW_CNT_MLL = 0000009E
RCBSW_CNT_XRE = 0000009C
RCBSW_ECLSEGS1Z = 0000007C
RCBSW_MAX_LNK = 00000058
RCBSW_MCOINT = 00000054
RCBSW_TIM_CNI = 00000076
RCBSW_TRANS = 0000000C
REASON_C_LENGTH = 00000006
REASON_W_DR = 00000000
REASON_W_MBX = 00000004
REASON_W_SS = 00000002
REPLY = 00000AA0
SCH\$GL_PCBVEC *****
SCH\$WAKE *****
SETUP_XWB = 0000064F
SIZ... = 00000001
SS\$ ABORT = 0000002C
SS\$ ACCVIO = 0000000C
SS\$ BADPARAM = 00000014
SS\$ CONNECFail = 000020DC
SS\$ DEVALLOC = 00000840
SS\$ FILNOTACC = 000000AC
SS\$ ILLIOFUNC = 000000F4
SS\$ INVLOGIN = 0000209C
SS\$ LINKABORT = 000020E4
SS\$ LINKDISCON = 000020EC
SS\$ LINKEXIT = 000020F4
SS\$ NOLINKS = 0000027C

R 03
R 03

G 03
G 03
G 03
R 03
X 03
X 03
R 03

NETDRVSES
Symbol table

K 1

- DECnet Session Control Module for NETD 16-SEP-1984 01:32:10 VAX/VMS Macro V04-00 Page 75
5-SEP-1984 02:20:26 [NETACP.SRC]NETDRVSES.MAR;1 (59)

```

SS$NOMBX      = 00000274
SS$NORMAL     = 00000001
SS$NOSUCHNODE = 0000028C
SS$NOSUCHOBJ  = 000020A4
SS$PATHLOST   = 000020FC
SS$PROTOCOL   = 00002074
SS$REJECT     = 00000294
SS$REMRSRC    = 0000206C
SS$SHUT       = 0000208C
SS$THIRDPARTY = 0000207C
SS$UNREACHABLE = 00002094
TQ$B_RQTYPE   = 0000000B
TQ$C_SSREPT   = 00000005
TQ$C_FPC      = 0000000C
TQ$C_FR4      = 00000014
TQ$C_DELTA    = 00000020
TQ$C_REPEAT   = 00000002
TR$C_MAXHDR   = 0000001C
TR$C_NI_ALLEND1 = 040000AB
TR$C_NI_ALLEND2 = 00000000
TR$C_NI_ALLROU1 = 030000AB
TR$C_NI_ALLROU2 = 00000000
TR$C_NI_PREFIX = 000400AA
TR$C_NI_PROT   = 00000360
TR$C_PRI_ECL   = 0000001F
TR$C_PRI_RTHRU = 0000001F
TR$UPDATE     = *****
TR3$$$QUAL_MSG = 00000000
TR3$$$QUAL_RTFLG = 00000000
TR3$C_HSZ_DATA = 00000006
TR3$C_MSG_DATA = 00000002
TR3$C_MSG_HELLO = 00000005
TR3$C_MSG_INIT = 00000001
TR3$C_MSG_NOP2 = 00000008
TR3$C_MSG_ROUT = 00000007
TR3$C_MSG_STR2 = 00000058
TR3$C_MSG_VERF = 00000003
TR3$M_MSG_CTL  = 000C0001
TR3$M_MSG_RTH  = 00000002
TR3$M_RTFLG_PH2 = 00000040
TR3$M_RTFLG_RQR = 00000008
TR3$M_RTFLG_RTS = 00000010
TR3$R_QUAL     = 00000000
TR3$S_QUAL     = 00000001
TR3$S_QUAL_MSG = 00000001
TR3$S_QUAL_RTFLG = 00000001
TR3$S_RTFLG_012 = 00000003
TR3$S_TR3MSG   = 00000001
TR3$V_MSG_CTL  = 00000000
TR3$V_MSG_RTH  = 00000001
TR3$V_RTFLG_012 = 00000000
TR3$V_RTFLG_5  = 00000005
TR3$V_RTFLG_7  = 00000007
TR3$V_RTFLG_PH2 = 00000006
TR3$V_RTFLG_RQR = 00000003
TR3$V_RTFLG_RTS = 00000004
TR4$$$QUAL_ADDR = 00000000

```

x 03

```

TR4$$$QUAL_RTFLG = 00000000
TR4$$$QUAL_SCLASS = 00000000
TR4$C_BCE_MID1   = 040000AB
TR4$C_BCE_MID2   = 00000000
TR4$C_BCR_MID1   = 030000AB
TR4$C_BCR_MID2   = 00000000
TR4$C_BCT3MULT   = 00000008
TR4$C_END_NODE   = 00000003
TR4$C_HIORD      = 000400AA
TR4$C_HSZ_DATA   = 00000015
TR4$C_MSG_BCEHEL = 0000000D
TR4$C_MSG_BCRHEL = 0000000B
TR4$C_MSG_LDATA  = 00000006
TR4$C_MSG_RDATA  = 00000002
TR4$C_PRO_TYPE   = 00000360
TR4$C_RTR_LVL1   = 00000002
TR4$C_RTR_LVL2   = 00000001
TR4$C_T3MULT     = 00000002
TR4$C_VER_HIB    = 00000000
TR4$C_VER_LOWW   = 00000002
TR4$M_ADDR_AREA  = 0000FC00
TR4$M_ADDR_DEST  = 000003FF
TR4$M_RTFLG_INI  = 00000020
TR4$M_RTFLG_LNG  = 00000004
TR4$M_RTFLG_RQR  = 00000008
TR4$M_RTFLG_RTS  = 00000010
TR4$R_QUAL       = 00000000
TR4$S_ADDR_AREA  = 00000006
TR4$S_ADDR_DEST  = 0000000A
TR4$S_QUAL       = 00000002
TR4$S_QUAL_ADDR  = 00000002
TR4$S_QUAL_RTFLG = 00000001
TR4$S_QUAL_SCLASS = 00000001
TR4$S_RTFLG_01   = 00000002
TR4$S_RTFLG_VER  = 00000002
TR4$S_SCLASS_57  = 00000003
TR4$S_TR4MSG     = 00000002
TR4$V_ADDR_AREA  = 0000000A
TR4$V_ADDR_DEST  = 00000000
TR4$V_RTFLG_01   = 00000000
TR4$V_RTFLG_INI  = 00000005
TR4$V_RTFLG_LNG  = 00000002
TR4$V_RTFLG_RQR  = 00000003
TR4$V_RTFLG_RTS  = 00000004
TR4$V_RTFLG_VER  = 00000006
TR4$V_SCLASS_1   = 00000001
TR4$V_SCLASS_57  = 00000005
TR4$V_SCLASS_BC  = 00000004
TR4$V_SCLASS_LS  = 00000002
TR4$V_SCLASS_METR = 00000000
TR4$V_SCLASS_SUBA = 00000003
UCB$B_DIPL       = 0000005E
UCB$B_FIPL       = 0000000B
UCB$C_LENGTH     = 00000090
UCB$L_AMB        = 00000060
UCB$L_DDB        = 00000028
UCB$L_DEVCHAR    = 00000038

```

NETDRVSES
Symbol table

- DECnet Session Control Module for NETD L 1
16-SEP-1984 01:32:10 VAX/VMS Macro V04-00
5-SEP-1984 02:20:26 [NETACP.SRC]NETDRVSES.MAR;1

Page 76
(59)

NET
V04

UCBSL_DEVDEPEND	=	00000044		
UCBSL_IOQFL	=	0000004C		
UCBSL_IRP	=	00000058		
UCBSL_LINK	=	00000030		
UCBSL_VCB	=	00000034		
UCBSM_BSY	=	00000100		
UCBSM_ONLINE	=	00000010		
UCBSM_TEMPLATE	=	00002000		
UCBSV_BSY	=	00000008		
UCBSW_DEVBUSIZ	=	00000042		
UCBSW_MB_SEED	=	00000000		
UCBSW_STS	=	00000064		
UCBSW_UNIT	=	00000054		
UNKNOWN	=	00000B2D	R	03
VECSL_ADP	=	00000014		
VECSL_INITIAL	=	0000000C		
VECSL_START	=	0000001C		
VECSL_UNITINIT	=	00000018		
XWB	=	00000000		
XWBSS	=	00000160	G	
XWBSB_ACCESS	=	0000000B		
XWBSB_ADJ_INX	=	00000124	G	
XWBSB_DATA	=	0000005B		
XWBSB_FIPL	=	0000001F		
XWBSB_LOGIN	=	000000CC		
XWBSB_LPRNAM	=	000000A4		
XWBSB_PRO	=	0000005A		
XWBSB_RID	=	0000006F		
XWBSB_RPRNAM	=	000000B8		
XWBSB_SP3	=	0000006E		
XWBSB_STA	=	0000001E		
XWBSB_TYPE	=	0000000A		
XWBSB_X_FLW	=	0000006C		
XWBSB_X_FLWCNT	=	0000006D		
XWBSB_COMLNG	=	000000A4		
XWBSB_CONLNG	=	00000112		
XWBSB_DATA	=	00000010		
XWBSB_LOGIN	=	00000040		
XWBSB_LPRNAM	=	00000014		
XWBSB_NDC_LNG	=	00000020		
XWBSB_NUMSTA	=	00000008		
XWBSB_RID	=	00000010		
XWBSB_RPRNAM	=	00000014		
XWBSB_STA_CAR	=	00000002		
XWBSB_STA_CCS	=	00000004		
XWBSB_STA_CIR	=	00000003		
XWBSB_STA_CIS	=	00000001		
XWBSB_STA_CLO	=	00000000		
XWBSB_STA_DIR	=	00000006		
XWBSB_STA_DIS	=	00000007		
XWBSB_STA_RUN	=	00000005		
XWBSL_DEA_IRP	=	00000104		
XWBSL_FPC	=	00000020		
XWBSL_FR3	=	00000024		
XWBSL_FR4	=	00000028		
XWBSL_ICB	=	00C0010C		
XWBSL_IRP_ACC	=	00000080		

XWBSL_LINK	=	0000002C		
XWBSL_ORGUCB	=	00000010		
XWBSL_PID	=	00000034		
XWBSL_PTR_RTND	=	00000120	G	
XWBSL_VCB	=	00000030		
XWBSL_WLBL	=	00000004		
XWBSL_WLFL	=	00000000		
XWBSM_FLG_BREAK	=	00000001		
XWBSM_FLG_CLO	=	00000200		
XWBSM_FLG_I AVL	=	00001000		
XWBSM_FLG_SCD	=	00000100		
XWBSM_FLG_SDACK	=	00000008		
XWBSM_FLG_SDFL	=	00004000		
XWBSM_FLG_SDT	=	00000080		
XWBSM_FLG_SIACK	=	00000004		
XWBSM_FLG_SIFL	=	00002000		
XWBSM_FLG_SLI	=	00000010		
XWBSM_FLG_TBPR	=	00000800		
XWBSM_FLG_WBP	=	00000040		
XWBSM_FLG_WBUF	=	00000002		
XWBSM_FLG_WDAT	=	00000400		
XWBSM_FLG_WHGL	=	00000020		
XWBSM_FLG_WMSK	=	0000039D		
XWBSM_PRO_CCA	=	00000008		
XWBSM_PRO_NAR	=	00000010		
XWBSM_PRO_NFC	=	00000001		
XWBSM_PRO_PH2	=	00000004		
XWBSM_PRO_SFC	=	00000002		
XWBSM_STS_ASTPND	=	00000400		
XWBSM_STS_ASTREQ	=	00000800		
XWBSM_STS_CON	=	00000010		
XWBSM_STS_DIS	=	00000008		
XWBSM_STS_DTNAK	=	00000100		
XWBSM_STS_LINAK	=	00000200		
XWBSM_STS_NDC	=	00001000		
XWBSM_STS_OVF	=	00000080		
XWBSM_STS_RBP	=	00000040		
XWBSM_STS_SOL	=	00000004		
XWBSM_STS_TID	=	00000001		
XWBSM_STS_TLI	=	00000002		
XWBSM_STS_TMO	=	00000020		
XWBSQ_FORK	=	00000014		
XWBSQ_FREE_CXB	=	00000118		
XWBSR_CON_BLK	=	000000A4		
XWBSR_RUN_BLK	=	000000A4		
XWBS	=	00000006		
XWBS_COMLNG	=	0000006E		
XWBS_CON_BLK	=	0000006E		
XWBS_DATA	=	00000010		
XWBS_DT	=	00000030		
XWBS_FLG	=	00000002		
XWBS_FORK	=	00000008		
XWBS_FREE_CXB	=	00000008		
XWBS_LI	=	00000030		
XWBS_LOGIN	=	0000003F		
XWBS_LPRNAM	=	00000013		
XWBS_NDC	=	00000020		

NETDRVSES
Symbol table

M 1
- DECnet Session Control Module for NETD 16-SEP-1984 01:32:10 VAX/VMS Macro V04-00
5-SEP-1984 02:20:26 [NETACP.SRC]NETDRVSES.M, R;1

Page 77
(59)

NETI
V04.

XWBS\$PRO = 00000001
XWBS\$RID = 00000010
XWBS\$RPRNAM = 00000013
XWBS\$RUN_BLK = 00000064
XWBS\$STS = 00000002
XWBS\$XWB = 00000120
XWBS\$ = 00000112
XWBS\$DATA = 0000005C
XWBS\$DT = 000000A4
XWBS\$LI = 000000D4
XWBS\$LOGIN = 000000CD
XWBS\$LPRNAM = 000000A5
XWBS\$RID = 00000070
XWBS\$RPRNAM = 000000B9
XWBS\$TR3HDR = 00000158 G
XWBS\$V_FLG_BREAK = 00000000
XWBS\$V_FLG_CLO = 00000009
XWBS\$V_FLG_I AVL = 0000000C
XWBS\$V_FLG_SCD = 00000008
XWBS\$V_FLG_SDACK = 00000003
XWBS\$V_FLG_SDFL = 0000000E
XWBS\$V_FLG_SDT = 00000007
XWBS\$V_FLG_SIAK = 00000002
XWBS\$V_FLG_SIFL = 0000000D
XWBS\$V_FLG_SLI = 00000004
XWBS\$V_FLG_TBPR = 0000000B
XWBS\$V_FLG_WBP = 00000006
XWBS\$V_FLG_WBUF = 00000001
XWBS\$V_FLG_WDAT = 0000000A
XWBS\$V_FLG_WHGL = 00000005
XWBS\$V_PRO_CCA = 00000003
XWBS\$V_PRO_NAR = 00000004
XWBS\$V_PRO_NFC = 00000000
XWBS\$V_PRO_PH2 = 00000002
XWBS\$V_PRO_SFC = 00000001
XWBS\$V_STS_ASTPND = 0000000A
XWBS\$V_STS_ASTREQ = 0000000B
XWBS\$V_STS_CON = 00000004
XWBS\$V_STS_DIS = 00000003
XWBS\$V_STS_DTNK = 00000008
XWBS\$V_STS_LINAK = 00000009
XWBS\$V_STS_NDC = 0000000C
XWBS\$V_STS_OVF = 00000007
XWBS\$V_STS_RBP = 00000006
XWBS\$V_STS_SOL = 00000002
XWBS\$V_STS_TID = 00000000
XWBS\$V_STS_TLI = 00000001
XWBS\$V_STS_TMO = 00000005
XWBS\$CI_PATH = 00000110
XWBS\$DE[AY = 0000004E
XWBS\$DLY_FACT = 00000056
XWBS\$DLY_WGHT = 00000058
XWBS\$ELAPSE = 0000004A
XWBS\$FLG = 0000001C
XWBS\$LOCLNK = 0000003E
XWBS\$LOCSIZ = 00000040
XWBS\$PATH = 00000038

XWBS\$W_PROGRESS = 00000052
XWBS\$W_REFCNT = 0000000C
XWBS\$W_REMLNK = 0000003C
XWBS\$W_REMNOD = 0000003A
XWBS\$W_REMSIZ = 00000042
XWBS\$W_RETRAN = 00000054
XWBS\$W_R_REASON = 00000044
XWBS\$W_SIZE = 00000008
XWBS\$W_STS = 0000000E
XWBS\$W_TIMER = 00000050
XWBS\$W_TIM_ID = 00000048
XWBS\$W_TIM_INACT = 0000004C
XWBS\$W_X_REASON = 00000046
XWBS\$Z_NDC = 00000084
XWB_C_LEN = 0000017C
XWB_LOCLNK = 00000097 R C3
_SACT_DFLT = 00000000
_SACT_INDEX = 00000016
_SEVENT_INDEX = 00000015
_SMK = 00007DFF
_STMP = 00000120 G

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

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR
\$AB\$\$	00000057 (87.)	01 (1.)	NOPIC USR
\$\$\$105_PROLOGUE	0000008E (142.)	02 (2.)	NOPIC USR
\$\$\$115_DRIVER	00000D93 (3475.)	03 (3.)	NOPIC USR

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	26	00:00:00.09	00:00:00.71
Command processing	157	00:00:01.15	00:00:04.92
Pass 1	990	00:00:45.45	00:01:28.11
Symbol table sort	0	00:00:05.56	00:00:12.19
Pass 2	520	00:00:10.73	00:00:19.59
Symbol table output	0	00:00:00.67	00:00:00.95
Psect synopsis output	2	00:00:00.05	00:00:00.15
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1698	00:01:03.71	00:02:06.71

The working set limit was 2000 pages.
243296 bytes (476 pages) of virtual memory were used to buffer the intermediate code.
There were 180 pages of symbol table space allocated to hold 3212 non-local and 320 local symbols.
3029 source lines were read in Pass 1, producing 35 object records in Pass 2.
91 pages of virtual memory were used to define 70 macros.

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

Macro library name	Macros defined
-\$255\$DUA28:[SHRLIB]NMALIBRY.MLB;1	0
-\$255\$DUA28:[SHRLIB]EVCDEF.MLB;1	0
-\$255\$DUA28:[NETACP.OBJ]NETDRV.MLB;1	3
-\$255\$DUA28:[NETACP.OBJ]NET.MLB;1	10
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	30
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	10
TOTALS (all libraries)	53

3485 GETS were required to define 53 macros.

There were no errors, warnings or information messages.

MACRO/L:S=LIS\$:NETDRVSES/OBJ=OBJ\$:NETDRVSES MSRC\$:NETDRVSES/UPDATE-(ENH\$:NETDRVSES)+EXECML\$/LIB+LIB\$:NET/LIB+LIB\$:NETDRV/LIB+SHRLIB\$

0277 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400
401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500
501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600
601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700
701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800
801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900
901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000

NETDRVOR
LIS

NETDRVSE
LIS

NETDRVSP
LIS

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY