

Ps  
--  
NE

NE

NE

NE

SR

NNN		NNN	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	AAAAAAAAAA		CCCCCCCCCCCC	PPPPPPPPPP	
NNN		NNN	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	AAAAAAAAAA		CCCCCCCCCCCC	PPPPPPPPPP	
NNN		NNN	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	AAAAAAAAAA		CCCCCCCCCCCC	PPPPPPPPPP	
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	NNN	EEEEEEEEEEEE	TTT	AAA	AAA	CCC	PPPPPPPPPP	PPP
NNN	NNN	NNN	EEEEEEEEEEEE	TTT	AAA	AAA	CCC	PPPPPPPPPP	PPP
NNN	NNN	NNN	EEEEEEEEEEEE	TTT	AAA	AAA	CCC	PPPPPPPPPP	PPP
NNN		NNNNNN	EEE	TTT	AAAAAAAAAAAAAAAA	AAA	CCC	PPP	PPP
NNN		NNNNNN	EEE	TTT	AAAAAAAAAAAAAAAA	AAA	CCC	PPP	PPP
NNN		NNNNNN	EEE	TTT	AAAAAAAAAAAAAAAA	AAA	CCC	PPP	PPP
NNN		NNN	EEE	TTT	AAA	AAA	CCC	PPP	PPP
NNN		NNN	EEE	TTT	AAA	AAA	CCC	PPP	PPP
NNN		NNN	EEE	TTT	AAA	AAA	CCC	PPP	PPP
NNN		NNN	EEE	TTT	AAA	AAA	CCC	PPP	PPP
NNN		NNN	EEEEEEEEEEEEEEEE	TTT	AAA	AAA	CCC	PPPPPPPPPP	PPP
NNN		NNN	EEEEEEEEEEEEEEEE	TTT	AAA	AAA	CCCCCCCCCCCC	PPPPPPPPPP	PPP
NNN		NNN	EEEEEEEEEEEEEEEE	TTT	AAA	AAA	CCCCCCCCCCCC	PPPPPPPPPP	PPP

```

NN      NN  EEEEEEEEE  TTTTTTTTT  EEEEEEEEE  VV      VV  TTTTTTTTT  LL      000000  GGGGGGGG
NN      NN  EEEEEEEEE  TTTTTTTTT  EEEEEEEEE  VV      VV  TTTTTTTTT  LL      000000  GGGGGGGG
NN      NN  EE          TT          EE          VV      VV  TT          LL      00      00  GG
NN      NN  EE          TT          EE          VV      VV  TT          LL      00      00  GG
NNNN    NN  EE          TT          EE          VV      VV  TT          LL      00      00  GG
NNNN    NN  EE          TT          EE          VV      VV  TT          LL      00      00  GG
NN  NN  NN  EEEEEEEEE  TT          EE          VV      VV  TT          LL      00      00  GG
NN  NN  NN  EEEEEEEEE  TT          EE          VV      VV  TT          LL      00      00  GG
NN      NNNN EE          TT          EE          VV      VV  TT          LL      00      00  GG  GGGGGG
NN      NNNN EE          TT          EE          VV      VV  TT          LL      00      00  GG  GGGGGG
NN      NN  EE          TT          EE          VV      VV  TT          LL      00      00  GG  GG
NN      NN  EEEEEEEEE  TT          EEEEEEEEE  VV      VV  TT          LL      00      00  GG  GG
NN      NN  EEEEEEEEE  TT          EEEEEEEEE  VV      VV  TT          LL      00      00  GG  GG

```

```

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)	143	DECLARATIONS
(4)	280	Event timer action routine
(5)	300	Internal inbound raw event processing
(7)	645	Inbound raw event processing
(8)	771	STARTUP_EVL - Start EVL process
(9)	801	Event logging database changes
(10)	850	Outbound raw event processing
(11)	929	NET\$SET_CTR_TIMER - Reset automatic counter timer

```
0000 1 .TITLE NETEVTLOG - Process Event logging needs
0000 2 .IDENT 'V04-000'
0000 3 .DEFAULT DISPLACEMENT,WORD
0000 4
0000 5 :*****
0000 6 :*
0000 7 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 :* ALL RIGHTS RESERVED.
0000 10 :*
0000 11 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 :* TRANSFERRED.
0000 17 :*
0000 18 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 :* CORPORATION.
0000 21 :*
0000 22 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 :*
0000 25 :*
0000 26 :*****
0000 27 :
0000 28 :
0000 29 : FACILITY: NETWORK ACP
0000 30 :
0000 31 : ABSTRACT:
0000 32 :
0000 33 : This module performs the bulk of processing required to
0000 34 : take care of network event logging needs.
0000 35 :
0000 36 : ENVIRONMENT:
0000 37 :
0000 38 : MODE = KERNEL
0000 39 :
0000 40 : AUTHOR: Scott G. Davis, CREATION DATE: 03-JUL-1980
0000 41 :
0000 42 : MODIFIED BY:
0000 43 :
0000 44 : V015 RNG0015 Rod Gamache 18-Jun-1984
0000 45 : Log Data Base Re-used events.
0000 46 :
0000 47 : V014 TMH0014 Tim Halvorsen 28-Apr-1983
0000 48 : Make LDO use "Adjacent node" rather than
0000 49 : "Expected node".
0000 50 :
0000 51 : V013 RNG0013 Rod Gamache 21-Apr-1983
0000 52 : Save/Restore R1 in NET$DBC_EFI/ESI.
0000 53 :
0000 54 : V012 TMH0012 Tim Halvorsen 07-Apr-1983
0000 55 : Allow caller to specify that no REASON parameter is to
0000 56 : be logged on TPL events.
0000 57 :
```

0000	58	:	V011	TMH0C11	Tim Halvorsen	29-Mar-1983	
0000	59	:		Add "aborted service request".			
0000	60	:					
0000	61	:	V010	TMH0010	Tim Halvorsen	22-Dec-1982	
0000	62	:		Modify a number of events to log packet beginning			
0000	63	:		(first 16 bytes) rather than packet header, which			
0000	64	:		implies a Phase III route header.			
0000	65	:					
0000	66	:	V009	TMH0009	Tim Halvorsen	05-Nov-1982	
0000	67	:		Add code to suppress the area number in node			
0000	68	:		addresses, if area routing is being hidden.			
0000	69	:		Fix area reachability chage so that it reports			
0000	70	:		the source area, not the source node.			
0000	71	:					
0000	72	:	V008	TMH0008	Tim Halvorsen	16-Sep-1982	
0000	73	:		Add support for automatic line counters.			
0000	74	:					
0000	75	:	V007	TMH0007	Tim Halvorsen	27-Jul-1982	
0000	76	:		Add support to handle Phase IV events.			
0000	77	:		Increase size of event buffer to allow for a large			
0000	78	:		number of "node reachable" events at startup time.			
0000	79	:		Rewrite READ_EVENTS so that it correctly transfers			
0000	80	:		only as many WHOLE events as will fit into the caller's			
0000	81	:		buffer, and so that it correctly shifts the remaining			
0000	82	:		events in the event buffer correctly. The previous			
0000	83	:		code was returning partial event records to EVL, and			
0000	84	:		causing the number of bytes "left" to be incorrectly			
0000	85	:		computed to be a very small number, thus wasting most			
0000	86	:		of the event buffer.			
0000	87	:					
0000	88	:	V006	TMH0006	Tim Halvorsen	30-Jun-1982	
0000	89	:		Dynamically allocate event buffer, rather than having			
0000	90	:		it statically defined in impure own storage.			
0000	91	:		Remove all explicit addressing specifiers, and make			
0000	92	:		the default addressing = word for the entire module.			
0000	93	:					
0000	94	:	V005	TMH0005	Tim Halvorsen	12-Apr-1982	
0000	95	:		Get address of utility buffer from cell, rather than			
0000	96	:		referencing a statically defined location.			
0000	97	:		Fix STARTUP_EVL to queue a WQE to do the job, since			
0000	98	:		STARTUP_OBJ calls CNF action routines, some of which			
0000	99	:		allocate the CNF static temporary buffer. Unfortunately,			
0000	100	:		this is required because we may be logging counters while			
0000	101	:		having the static temporary buffer allocated (CNT does this).			
0000	102	:		Fix code to search database using FNDMIN operator to expect			
0000	103	:		that the matched CNF will be returned in R10.			
0000	104	:		Fix bug which prevented the node address from being shown			
0000	105	:		in the event display if there is no node name currently			
0000	106	:		associated with that address.			
0000	107	:		Fix bug in reporting of "packet format error" event which			
0000	108	:		showed garbage after "packet beginning" parameter.			
0000	109	:		Add formatting for "local node state change",			
0000	110	:		"locally initiated state change", and "remotely initiated			
0000	111	:		state change" events.			
0000	112	:					
0000	113	:	V03-04	ADE0025	A.Eldridge	01-Feb-1981	
0000	114	:		Remove parameter count in front of counter block.			

```
0000 115 : Fix database determination while processing the counter timer.
0000 116 :
0000 117 : V03-03 ADE0024 A.Eldridge 19-Jan-1981
0000 118 : Include the 'packet beginning' and not the 'packet header'
0000 119 : as part of the event data for circuit initialization failure
0000 120 : events.
0000 121 :
0000 122 : V03-02 ADE0023 Al Eldridge 30-Nov-1981
0000 123 : Added zero counter event.
0000 124 :
0000 125 : V03-01 Al Eldridge 01-Nov-1981
0000 126 : Upgrade to V3.0.0 Network management. The changes are
0000 127 : primarily related to the change to the new Circuit/Line
0000 128 : model of the datalink layer.
0000 129 :
0000 130 : V022 ADE0022 Al Eldridge 05-Sep-1980
0000 131 : Further fixes to counter logging.
0000 132 :
0000 133 : V021 TMH0021 Tim Halvorsen 04-Sep-1980
0000 134 : Pass null string as SYS$NET to EVL process. Preserve all
0000 135 : registers in NET$DBC_EF1,ESI. Remove temporary definition of
0000 136 : EVL$C_VMS_DBC (was decimal 2000, should be hex 2000) and use
0000 137 : $EVCDEF.
0000 138 :
0000 139 : V020 ADE0020 Al Eldridge 20-Aug-1980
0000 140 : Log internally detected events. Log counters.
0000 141 :
```

```

0000 143      .SBTTL  DECLARATIONS
0000 144      :
0000 145      : MACROS
0000 146      :
0000 147      $ADJDEF
0000 148      $MSGDEF
0000 149      $NETSYMDEF
0000 150      $NETUPDDEF
0000 151      $NFBDEF
0000 152      $CNFDEF
0000 153      $CNRDEF
0000 154      $PRVDEF
0000 155      $RCBDEF
0000 156      $NMADEF
0000 157      $EVCDEF
0000 158      $RAWDEF
0000 159      $WQDEF
0000 160
0000 161      :
0000 162      : EQUATED SYMBOLS:
0000 163      :
0000 164
000000C1 0000 165 NMASC_PTY_CM1 = 193      ; && until it gets added to $NMADEF
0000 166
0000 167
0000001A 0000 168 EVL_OBJ = 26      ; Event logger object number
00000005 0000 169 NET$C_EVTTHRESH = 5      ; Event threshold
02FAF080 0000 170 NET$C_EVTTIMER = 10*1000*1000*5 ; Timer constant
00001F40 0000 171 NET$C_EVTBUFLTH = 8000      ; Length of event buffer
00000020 0000 172 NET$C_LSTEVLTH = 32      ; Length of "lost event" event
00001F00 0000 173 NET$C_AVLBUFLTH = NET$C_EVTBUFLTH - ; Length for normal events
0000 174      - <2*NET$C_LSTEVLTH>
0000 175
0000 176      :
0000 177      : mailbox message mask definitions
0000 178      :
0000 179
00000001 0000 180 MBX$V_EVTAVL = 1      ; Mask bit for MSG$_EVTAVL
00000002 0000 181 MBX$V_EVTRCVCHG = 2      ; Mask bit for MSG$_EVTRCVCHG
00000003 0000 182 MBX$V_EVTXMTCHG = 3      ; Mask bit for MSG$_EVTXMTCHG

```

```

0000 184 ;
0000 185 ; OWN STORAGE:
0000 186 ;
00000000 187 .PSECT NET_IMPURE,WRT,NOEXE,LONG
0000 188
00000000 0000 189 CNX_PLI_OLDTIM: .LONG 0 ; Old CNF timer for PLI's
00000000 0004 190 CNX_CRI_OLDTIM: .LONG 0 ; Old CNF timer for CRI's
00000000 0008 191 CNX_NDI_OLDTIM: .LONG 0 ; Old CNF timer for NDI's
000C 192
01' 000C 193 EVT_B_FLAGS: .BYTE EVTSM_EVTAVL ; Allow immediate event message
000D 194
000D 195 $VIELD EVT,0,- ; Define the flags
000D 196 <-
000D 197 <EVTAVL,1,M>,- ; Flag implies MSG$ EVTAVL can be sent
000D 198 <LOSTEVENT,1,M>,- ; Flag implies "lost event" event occurred
000D 199 <DBCEVENT,1,M>,- ; Database change event logged
000D 200 <CST_PLI,1,M>,- ; Line counter suppression timer ticking
000D 201 <CST_CRI,1,M>,- ; Circuit counter suppression timer ticking
000D 202 <CST_NDI,1,M>,- ; Node counter suppression timer ticking
000D 203 >
0000000F 000D 205 EVT_W_THRESH: .BLKW 1 ; No. of events available
0000 000F 206 EVT_W_LOST: .WORD 0 ; # event bytes lost
0000 0011 207 EVT_W_PEAK: .WORD 0 ; Peak value of EVT_W_LOST
00000000 0013 208 BASE_TIME: .LONG 0 ; Base time for counter logging
0017 209
0017 210 .ALIGN LONG
00000000 0018 211 EVT_L_BUFFER: .LONG 0 ; Address of event buffer
00000000 001C 212 EVT_L_BUFPTR: .LONG 0 ; Ptr to next buffer location
0020 213
0020 214 LOST_EVENT: ; Block to hold "lost event"
001E' 0020 215 .WORD 10$-LOST_EVENT ; Length of event
0000002A 0022 216 .BLKQ 1 ; For time-stamp
0000 002A 217 .WORD EVC$C_NMA_LOS ; Event code
FF 002C 218 .BYTE -1 ; No source for this event
0000003E 002D 219 .BLKB 17 ; No event-ID
003E 220 10$:
003E 221
003E 222 DBC_EVENT: ; Block containing "DBC event"
001E' 003E 223 .WORD 10$-DBC_EVENT ; Length of event
00000048 0040 224 .BLKQ 1 ; For time-stamp
2000 0048 225 .WORD EVC$C_VMS_DBC ; Event code
FF 004A 226 .BYTE -1 ; No source for this event
0000005C 004B 227 .BLKB 17 ; No event-ID
005C 228 10$:
005C 229
005C 230 NET$AB_EVT_WQE:: ; Common WQE for event reporting
00000080 005C 231 .BLKB WQE$C_LENGTH
0080 232
0080 233
00000000 234 .PSECT NET_PURE, LONG, NOWRT, NOEXE
0000 235
0000 236
00000000 0000 237 CNX$B_SPARE = 0 ; Spare, reserved for future use
00000001 0000 238 CNX$B_TIM_SUP = 1 ; RCB suppression timer bit i.d.
00000002 0000 239 CNX$W_ID_CTM = 2 ; WQE timer REQIDT field and database i.d.
00000004 0000 240 CNX$L_COUNTER = 4 ; CNF field i.d. of counter string

```



```

00000008 0000 241      CNX$DEL_TIME   = 8      ; CNF field i.d. of delta timer value
0000000C 0000 242      CNX$ABS_TIME  = 12     ; CNF field i.d. of absolute timer value
00000010 0000 243      CNX$OLD_TIME  = 16     ; Ptr to oldest CNF absolute due time value
00000014 0000 244      CNX$CNR_PTR   = 20     ; Ptr to CNR pointer
00000018 0000 245      CNX$C_LENGTH = 24
          0000 246
          0000 247 CNX_PLI: ; PLI CNX
          00 0000 248      .BYTE 0 ; Spare
          03 0001 249      .BYTE evt$v_cst_pli ; Log datalink counter suppression timer id
          0001 0002 250      .WORD  evc$c_src_lin ; WQE REQIDT value for datalinks
          0004 251      .CNFFLD pli,s,cnt ; Datalink counter string field i.d.
          0008 252      .CNFFLD pli,l,lct ; Datalink counter timer field i.d.
          000C 253      .CNFFLD pli,l,cta ; Datalink absolute timer field i.d.
00000000' 0010 254      .LONG  cnx_pli_oldtim ; Due time of oldest CNFs
00000000' 0014 255      .LONG  net$gl_cnr_pli ; Address of CRI CNR pointer
          0018 256
          0018 257 CNX_CRI: ; CRI CNX
          00 0018 258      .BYTE 0 ; Spare
          04 0019 259      .BYTE  evt$v_cst_cri ; Log datalink counter suppression timer id
          0003 001A 260      .WORD  evc$c_src_cir ; WQE REQIDT value for datalinks
          001C 261      .CNFFLD cri,s,cnt ; Datalink counter string field i.d.
          0020 262      .CNFFLD cri,l,lct ; Datalink counter timer field i.d.
          0024 263      .CNFFLD cri,l,cta ; Datalink absolute timer field i.d.
00000004' 0028 264      .LONG  cnx_cri_oldtim ; Due time of oldest CNFs
00000000' 002C 265      .LONG  net$gl_cnr_cri ; Address of CRI CNR pointer
          0030 266
          0030 267 CNX_NDI: ; NDI CNX
          00 0030 268      .BYTE 0 ; Spa.
          05 0031 269      .BYTE  evt$v_cst_ndi ; Log node counter suppression timer id
          0000 0032 270      .WORD  evc$c_src_nod ; WQE REQIDT value for nodes
          0034 271      .CNFFLD ndi,s,cnt ; Node counter string field i.d.
          0038 272      .CNFFLD ndi,l,cti ; Node counter timer field i.d.
          003C 273      .CNFFLD ndi,l,cta ; Node absolute timer field i.d.
00000008' 0040 274      .LONG  cnx_ndi_oldtim ; Due time of oldest CNFs
00000000' 0044 275      .LONG  net$gl_cnr_ndi ; Address of NDI CNR pointer
          0048 276

```

```

00000000 278      .PSECT NET_CODE,NOWRT,LONG,EXE
0000      279
0000      280      .SBTTL Event timer action routine
0000      281      :+
0000      282      : EVT_TIMER - This routine is called when the event timer threshold expires.
0000      283      :
0000      284      : FUNCTIONAL DESCRIPTION:
0000      285      :
0000      286      : Set the EVENT AVAILABLE flag (NET$V_EVTAVL)
0000      287      :
0000      288      :-
0000      289      EVT_TIMER:
55      DD 0000 290      PUSHL R5 ; Save timer block address
01      88 0002 291      BISB2 #EVT$M_EVTAVL,- ; Set the flag
000C'CF 0004 292      EVT_B_FLAGS
000D'CF B5 0007 293      TSTW EVT_W_THRESH ; Any events?
03      13 000B 294      BEQL 10$ ; If EQL no msgs, yet
0378    30 000D 295      BSBW SEND_EVT_MSG ; Send MBX MSG
50      8ED0 0010 296 10$: POPL R0 ; Recover timer block
FFEA'   30 0013 297      BSBW WQES$DEALLOCATE ; Deallocate it
05      0016 298      RSB ; Done

```

```

0017 300      .SBTTL Internal inbound raw event processing
0017 301      :+
0017 302      : NET$EVT_INTRAW - Process raw event detected internally
0017 303      :
0017 304      : FUNCTIONAL DESCRIPTION:
0017 305      :
0017 306      : A raw event is passed internally via a WQE. It is formatted and put into
0017 307      : the event buffer.
0017 308      :
0017 309      : INPUTS:      R11      CNR pointer as appropriate
0017 310      :           R10      CNF pointer as appropriate
0017 311      :           R9-R7     Scratch
0017 312      :           R6       LPD pointer if datalink event
0017 313      :           XWB     pointer if logical link event
0017 314      :           else srcatch
0017 315      :           R5       WQE pointer if appropriate
0017 316      :
0017 317      : OUTPUTS:     All registers are preserved
0017 318      :
0017 319      :-
0017 320      NET$EVT_INTRAW::      : Process internal raw event
0017 321      -BSBW NET$GETUTLBUF   : Get permission to use the utility
001A 322      :           : buffer (co-routine call)
001A 323      :
001A 324      PUSHR #*M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
001E 325      :
53 02 0000'CF C1 001E 326      ADDL3 NET$GL UTLBUF,#2,R3      : Setup output pointer
83 83 00000000'GF 7D 0024 327      MOVQ G^EXE$GQ SYSTIME,(R3)+      : Enter standard quadword time
      50 1C A5 3C 002B 328      MOVZWL WQE$W EVC_CODE(R5),R0      : Get the raw event code
      83 50 B0 002F 329      MOVW R0,(R3)+      : Enter the code
      17 10 0032 330      BSBB 50$      : Dispatch to complete building the
      : event
      : If LBC then abort logging
58 0F 50 E9 0034 332      BLBC R0,40$
57 53 58 C3 0037 333      MOVL NET$GL UTLBUF,R8      : Get original output pointer
      68 57 B0 0040 334      SUBL3 R8,R3,R7      : Calculate the data length
      0290 30 0043 335      MOVW R7,(R8)      : Store as the length field
      : Stuff it into the event buffer
OFFF 8F BA 0046 336      BSBB INTERNAL_EVENT
      :
OFFF 8F 05 0046 337      POPR #*M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
      :
      : 40$:
      : 50$:
      : Dispatch to finish building the event. The $DISPATCH skip chain is
      : used instead of one large $DISPATCH since the index codes are closely
      : packed within a class but widely separated from one class to another.
      :
004B 338      $DISPATCH R0,-
004B 339      <-
004B 340      :
004B 341      <EVC$C_TPL_APL, NON_PKT>, -; Aged packet loss
004B 342      <EVC$C_TPL_UPL, CIR_PKT>, -; Unreachable packet loss
004B 343      <EVC$C_TPL_RPL, CIR_PKT>, -; Out-of-range packet loss
004B 344      <EVC$C_TPL_OPL, CIR_PKT>, -; Oversized packet loss
004B 345      <EVC$C_TPL_PFM, CIR_BEG>, -; Packet format error
004B 346      <EVC$C_TPL_PRU, PRUS>, -; Partial routing update loss
004B 347      <EVC$C_TPL_VFR, VFR>, -; Verification reject
004B 348      <EVC$C_TPL_LDO, LDO>, -; Circuit down, operator fault
004B 349      <EVC$C_TPL_LDS, LDS>, -; Circuit down, software fault
004B 350
004B 351
004B 352
004B 353
004B 354
004B 355
004B 356

```

```

004B 357 <EVCSC_TPL_LDF, LDF> -; Circuit down, Circuit fault
004B 358 <EVCSC_TPL_LUP, CIR_ADJ> -; Circuit up
004B 359 <EVCSC_TPL_IOF, IOFS> -; Init failed, operator fault
004B 360 <EVCSC_TPL_ISF, ISF> -; Init failed, software fault
004B 361 <EVCSC_TPL_ILF, ILF> -; Init failed, Circuit fault
004B 362 <EVCSC_TPL_RCH, RCH> -; Node reachability change
004B 363 <EVCSC_TPL_AUP, CIR_ADJ> -; Adjacency up
004B 364 <EVCSC_TPL_ARJ, CIR_ADJ> -; Adjacency rejected
004B 365 <EVCSC_TPL_ACH, ACHS> -; Area reachability change
004B 366 >
0079 367
0079 368 $DISPATCH RO,-
0079 369 <-
0079 370 <EVCSC_NMA_CTR, COUNTER> -; Automatic counter timer
0079 371 <EVCSC_NMA_ZER, COUNTER> -; NCP ZERO counters command
0079 372 <EVCSC_NMA_ABS, ABS> -; Aborted service request
0079 373 >
0083 374
0083 375 $DISPATCH RO,-
0083 376 <-
0083 377 <EVCSC_NSL_DBR, COUNTER> -; Data base re-used event
0083 378 >
008B 379
008B 380 $DISPATCH RO,-
008B 381 <-
008B 382 <EVCSC_SCL_LNS, LNS> -; Local node state change
008B 383 >
0093 384
0093 385 $DISPATCH RO,-
0093 386 <-
0093 387 <EVCSC_DLL_LSC, LSC> -; Locally initiated state change
0093 388 <EVCSC_DLL_RSC, RSC> -; Remotely initiated state change
0093 389 >
50 D4 009D 390 CLRL RO ; Event unknown
05 009F 391 RSB ; Done
00A0 392
00A0 393 COUNTER:
50 1E A5 9A 00A0 394 MOVZBL WQESB_EVL_DT1(R5),RO ; Get counter database i.d.
00A4 395 $DISPATCH RO,- ; Dispatch on database type
00A4 396 <-
00A4 397 <EVCSC_SRC_NOD, NOD_COU>,- ; Log and clear node counters
00A4 398 <EVCSC_SRC_CIR, CIR_COU>,- ; Log and clear circuit counters
00A4 399 <EVCSC_SRC_LIN, LIN_COU>,- ; Log and clear line counters
00A4 400 >
50 D4 00B0 401 CLRL RO ; Database unknown
05 00B2 402 RSB ; Done
00B3 403
00B3 404 NOD_COU: ; Node counters
011B 30 00B3 405 BSBW ENTER_SRCNOD ; Enter source node i.d.
08 11 00B6 406 BRB COU ; Enter the counters
00B8 407 CIR_COU: ; Circuit counters
013A 30 00B8 408 BSBW ENTER_SRCCIR ; Enter Circuit i.d.
03 11 00BB 409 BRB COU ; Enter the counters
00BD 410 LIN_COU: ; Line counters
0154 30 00BD 411 BSBW ENTER_SRCLIN ; Enter Line ID
00C0 412
00C0 413 COU: ; Log and clear the counters

```

63	50	1F	55	DD	00C0	414	PUSHL	R5	:	Save reg
	18	B5	A5	9A	00C2	415	MOVZBL	WQESB_EVL_DT2(R5),R0	:	Get length of counter block
			50	28	00C6	416	MOVCS	R0,-	:	
					00CB	417		@WQESL_EVL_PKT(R5),(R3)	:	Move the counter block
			50	01	00CB	418	MOVL	#1,R0	:	Indicate success
				55	8ED0	419	POPL	R5	:	Recover WQE pointer
					00D1	420	RSB		:	
					00D2	421			:	
					00D2	422	NON_PKT:		:	No source, packet header
					00D2	423	BSBW	ENTER_NO_SRC	:	Enter null source field
					0148	31	BRW	ENTER_PKTHDR	:	Enter the packet header
					00D8	425			:	
					00D8	426	CIR_PKT:		:	Circuit source, adj, packet header
					011A	30	BSBW	ENTER_SRCCIR	:	Enter the source Circuit
					83	08	MOVW	#EVCSC_TPL_PADJ,(R3)+	:	Identify next field
					0181	30	BSBW	PNA_NODE	:	Enter partner node id
					013C	31	BRW	ENTER_Pk HDR	:	Enter the packet header
					00E4	431			:	
					00E4	432	CIR_BEG:		:	Circuit source, adj, packet beginning
					010E	30	BSBW	ENTER_SRCCIR	:	Enter the source Circuit
					83	08	MOVW	#EVCSC_TPL_PADJ,(R3)+	:	Identify next field
					0175	30	BSBW	PNA_NODE	:	Enter partner node id
					0159	31	BRW	ENTER_PPKB	:	Enter packet beginning
					00F0	437			:	
					00F0	438	PRU.		:	Partial routing update loss
					0102	30	BSBW	ENTER_SRCCIR	:	Enter source Circuit
					0153	30	BSBW	ENTER_PPKB	:	Enter the packet header
					83	02	MOVW	#EVCSC_TPL_PHIA,(R3)+	:	Identify next field
					83	02	MOVB	#NMASC_PTY_DU2,(R3)+	:	Identify field format
					00FC	443			:	
					00FC	444	ASSUME	WQESB_EVL_DT2-WQESB_EVL_DT1 EQ 1	:	
					00FC	445			:	
					83	1E	MOVW	WQESB_EVL_DT1(R5),(R3)+	:	Enter partner's highest
									:	reachable node address
					83	08	MOVW	#EVCSC_TPL_PADJ,(R3)+	:	Identify adjacent node
					015C	30	BSBW	PNA_NODE	:	Enter partner node id
					50	01	MOVB	#1,R0	:	Success
						05	RSB		:	
					010A	452	VFR:		:	Verification reject
					00E8	30	BSBW	ENTER_SRCCIR	:	Enter the source Circuit
					83	03	MOVW	#EVCSC_TPL_PNOD,(R3)+	:	Identify next field
					014F	31	BRW	PNA_NODE	:	Enter partner node id
					0113	456			:	
					0113	457	IOF:		:	Init failure, operator fault
									:	Same as ISF, except add:
					83	23	BSBB	ISF	:	Identify next field (version)
					83	06	MOVW	#EVCSC_TPL_PVRS,(R3)+	:	Enter format type
					83	C3	MOVB	#NMASC_PTY_CM3,(R3)+	:	Enter format type
					83	01	MOVB	#NMASC_PTY_DU1,(R3)+	:	Enter format type
					83	0000	MOVB	NET\$GL_INITVER,(R3)+	:	Enter version number
					83	01	MOVB	#NMASC_PTY DU1,(R3)+	:	Enter format type
					83	0001	MOVB	NET\$GL_INITVER+1,(R3)+	:	Enter ECO number
					83	01	MOVB	#NMASC_PTY DU1,(R3)+	:	Enter format type
					83	0002	MOVB	NET\$GL_INITVER+2,(R3)+	:	Enter user ECO number
					50	01	MOVB	#1,R0	:	Success
						05	RSB		:	
							ISF:		:	Init failure, software fault
							BSBB	CIR_REASON	:	Enter circuit id, reason
					17	10			:	
					0138	469			:	
					0138	470			:	

```

010C 31 013A 471 BRW ENTER_PPKB ; Enter packet header
      013D 472
      013D 473 LDO:
      013D 474 LDS: ; Adjacency forced down by software
83 12 10 013D 475 BSBB CIR_REASON ; Enter common header
      08 B0 013F 476 MOVW #EVC$C_TPL_PADJ,(R3)+ ; Identify next field
      011D 30 0142 477 BSBW PNA_NODE ; Enter partner node id
      0101 31 0145 478 BRW ENTER_PPKB ; Enter packet header
      0148 479
      0148 480 CIR_ADJ: ; Enter circuit id, adjacent node
83 00AA 30 0148 481 BSBW ENTER_SRCCIR ; Enter source Circuit id
      08 B0 014B 482 MOVW #EVC$C_TPL_PADJ,(R3)+ ; Identify adjacent node
      0111 31 014E 483 BRW PNA_NODE ; Enter partner node id
      0151 484
      0151 485 ILF: ; Init failure, circuit fault
      0151 486 LDF: ; Circuit failure, Circuit fault
      0151 487 CIR_REASON: ; Enter circuit id, reason code
      00A1 30 0151 488 BSBW ENTER_SRCCIR ; Enter source Circuit id
      1E A5 95 0154 489 TSTB WQESB_EVL_DT1(R5) ; Any reason specified?
      05 19 0157 490 BLSS 90$ ; Exit if not
83 05 B0 0159 491 MOVW #EVC$C_TPL_PRSN,(R3)+ ; Identify next field
      12 11 015C 492 BRB CD1 ; Enter field's value
      50 01 90 015E 493 90$: MOVVB #1,R0 ; Signal success
      05 0161 494 RSB
      0162 495
      0162 496 RCH: ; Node reachability change
83 006C 30 0162 497 BSBW ENTER_SRCNOD ; Enter the source node
      07 B0 0165 498 MOVW #EVC$C_TPL_PSTS,(R3)+ ; Identify next field
      06 11 0168 499 BRB CD1
      016A 500
      016A 501 ALh: ; Area reachability change
83 004E 30 016A 502 BSBW ENTER_SRCAREA ; Enter the source area
      07 B0 016D 503 MOVW #EVC$C_TPL_PSTS,(R3)+ ; Identify next field
      0170 504
83 81 8F 90 0170 505 CD1: MOVVB #NMASC_PTY_CD1,(R3)+ ; Enter field format type
83 1E A5 90 0174 506 MOVVB WQESB_EVL_DT1(R5),(R3)+ ; Enter qualifying data byte
      50 01 90 0178 507 MOVVB #1,R0 ; Signal success
      05 017B 508 RSB
      017C 509
83 0034 30 017C 510 LNS: BSBW ENTER_NO_SRC ; Enter no source ID
      83 81 8F B0 017F 511 MOVW #EVC$C_SCL_PRSN,(R3)+ ; Enter "reason" parameter type
83 18 A5 90 0182 512 MOVVB #NMASC_PTY_CD1,(R3)+ ; Enter field format type
      83 01 B0 0186 513 MOVVB WQESL_EVL_PKT(R5),(R3)+ ; Enter reason code
      E1 10 018A 514 MOVW #EVC$C_SCL_POLD,(R3)+ ; Enter "old state" parameter type
      83 02 B0 018D 515 BSBB CD1 ; Enter coded byte from DT1
      83 81 8F B0 018F 516 MOVW #EVC$C_SCL_PNEW,(R3)+ ; Enter "new state" parameter type
83 1F A5 90 0192 517 CD1_2: MOVVB #NMASC_PTY_CD1,(R3)+ ; Enter field format type
      50 01 90 0196 518 MOVVB WQESB_EVL_DT2(R5),(R3)+ ; Enter qualifying data byte
      05 019A 519 MOVVB #1,R0 ; Signal success
      019D 520 RSB
      019E 521
      019E 522 LSC:
83 0054 30 019E 523 RSC: BSBW ENTER_SRCCIR ; Enter source circuit
      00 B0 01A1 524 MOVW #EVC$C_DLL_POLD,(R3)+ ; Enter "old state" parameter type
      CA 10 01A4 525 BSBB CD1 ; Enter coded byte from DT1
      83 01 B0 01A6 526 MOVW #EVC$C_DLL_PNEW,(R3)+ ; Enter "new state" parameter type
      E7 11 01A9 527 BRB CD1_2 ; Enter coded byte from DT2; and exit

```

			01AB	528					
			01AB	529	ABS:				: "Aborted service request"
			01AB	530					: Enter circuit id, reason code
83	0047	30	01AB	531		BSBW	ENTER SRCCIR		: Enter source Circuit id
	03	B0	01AE	532		MOVW	#EVCSC_NMA_PRSN,(R3)+		: Identify next field
	BD	11	01B1	533		BRB	CD1		: Enter field's value

				01B3	535	ENTER_NO SRC:			: Enter null source field
	83	FF	8F	90	01B3	536	MOVB	#EVCSC_SRC_NON,(R3)+	: No source
			57	D4	01B7	537	CLRL	R7	: Init count field
			47	11	01B9	538	BRB	ENT_17	: Zero the source field
					01BB	539			
					01BB	540	ENTER_SRCAREA:		: Enter source area
	83	05		90	01BB	541	MOVB	#EVCSC_SRC_ARE,(R3)+	: Enter source type
	83	12	A5	90	01BE	542	MOVB	WQESW_REQIDT(R5),(R3)+	: Store the area number
			55	DD	01C2	543	PUSHL	R5	: Save registers
63	10	00	6E	00	2C	01C4	MOVCS	#0,(SP),#0,#16,(R3)	: Zero rest of 17 byte fixed field
			55	8ED0	01CA	545	POPL	R5	: Restore registers
			50	01	DD	01CD	MOVL	#1,R0	: Success
				05	01D0	547	RSB		
					01D1	548			
					01D1	549	ENTER_SRCNOD:		: Enter source node
	83	00		90	01D1	550	MOVB	#EVCSC_SRC_NOD,(R3)+	: Enter source type
	51	12	A5	3C	01D4	551	MOVZWL	WQESW_REQIDT(R5),R1	: Get the node address
			09	12	01D8	552	BNEQ	10\$	: Branch if not local node
50	0000	'CF		DD	01DA	553	MOVL	NET\$GL_PTR_VCB,R0	: Get the RCB address
	51	OE	A0	3C	01DF	554	MOVZWL	RCBSW_ADDR(R0),R1	: Enter the local node address
		FE1A'		30	01E3	555	10\$: BSBW	SUPPRESS_AREA	: Suppress area, if necessary
	83	51		B0	01E6	556	MOVW	R1,(R3)+	: Enter the node address
					01E9	557	\$CNFFLD	ndi,s,na,R9	: Identify the node name field
			0D	10	01F0	558	BSBB	ENT_SRC	: Enter padded node name
			73	B5	01F2	559	TSTW	-(R3)	: Backup two bytes to account for
				05	01F4	560	RSB		: node address at begining in order
					01F5	561			: to keep a total of 17 bytes
					01F5	562			
					01F5	563	ENTER_SRCCIR:		: Enter source Circuit id
	83	03		90	01F5	564	MOVB	#EVCSC_SRC_CIR,(R3)+	: Enter source type
					01F8	565	\$CNFFLD	cri,s,na,r9	: Get the Circuit name field i.d.
		FDFE'		30	01FF	566	ENT_SRC: BSBW	CNF\$GET_FIELD	: Get the source i.d. name
			55	DD	0202	567	ENT_17: PUSHL	R5	: Save critical reg
	83	57		90	0204	568	MOVB	R7,(R3)+	: Enter length of name
63	10	00	68	57	2C	0207	MOVCS	R7,(R8),#0,#16,(R3)	: Enter the name
			55	8ED0	020D	570	POPL	R5	: Restore reg
			50	01	90	0210	MOVW	#1,R0	: Success
				05	0213	572	RSB		
					0214	573			
					0214	574	ENTER_SRCLIN:		: Enter source Line id
	83	01		90	0214	575	MOVB	#EVCSC_SRC_LIN,(R3)+	: Enter source type
					0217	576	\$CNFFLD	pli,s,na,r9	: Get the Line name field i.d.
		DF		11	021E	577	BRB	ENT_SRC	: Store the parameter value
					0220	578			
					0220	579	ENTER_PKTHDR:		: Get msg pointer
	50	18	A5	DD	0220	580	MOVL	WQESL_EVL_PKT(R5),R0	: Skip if none
			1F	13	0224	581	BEQL	90\$	: Enter field i.d.
	83	00		B0	0226	582	MOVW	#EVCSC_TPL_PPKH,(R3)+	: Format type for multiple field
	83	C4	8F	90	0229	583	MOVB	#NMASC_PTY_CM4,(R3)+	: Format type for message flags
	83	21		90	022D	584	MOVB	#NMASC_PTY_H1,(R3)+	: Enter message flags
	83	80		90	0230	585	MOVB	(R0)+(R3)+	: Format type for dst node
	83	02		90	0233	586	MOVB	#NMASC_PTY_DU2,(R3)+	: Enter dst node address
	83	80		B0	0236	587	MOVW	(R0)+(R3)+	: Format type for src node
	83	02		90	0239	588	MOVB	#NMASC_PTY_DU2,(R3)+	: Enter src node address
	83	80		B0	023C	589	MOVW	(R0)+(R3)+	: Format type for visits field
	83	21		90	023F	590	MOVB	#NMASC_PTY_H1,(R3)+	: Enter visits field
	83	80		90	0242	591	MOVB	(R0)+(R3)+	





```

02CC 645 .SBTTL Inbound raw event processing
02CC 646 :+
02CC 647 : NET$LOG_EVENT - Put a raw event into the event buffer
02CC 648 :
02CC 649 : FUNCTIONAL DESCRIPTION:
02CC 650 :
02CC 651 : A raw event is passed to NETACP. If a "lost event" event is already in
02CC 652 : the raw event buffer, then the operation is ignored. If there is no more
02CC 653 : room for events, the "lost event" event is placed in the buffer and the
02CC 654 : flag is set to so indicate. If an event is placed in the buffer, and the
02CC 655 : EVTAVL flag is set, then a mailbox message (MSG$_EVTAVL) is broadcast.
02CC 656 : Events put into the buffer are time-stamped.
02CC 657 :
02CC 658 : INPUTS: NET$GL_SIZ_P2 - size of input event
02CC 659 : NET$GL_PTR_P2 - address of input event
02CC 660 :
02CC 661 : OUTPUTS: MBX message may be broadcast (MSG$_EVTAVL)
02CC 662 : RO - Status
02CC 663 :
02CC 664 : -
02CC 665 : .ENABL LSB
02CC 666 :
02CC 667 NET$LOG_EVENT::
57 0000'CF DO 02CC 668 : NET$LOG_EVENT:: : Entry point
58 0000'CF DO 02D1 669 : MOVL NET$GL_SIZ_P2,R7 : Get no. of bytes in event
: MOVL NET$GL_PTR_P2,R8 : Get address of event data
02D6 670
02D6 671 INTERNAL_EVENT: : Local entry point
68 57 B1 02D6 672 : CMPW R7,(R8) : Counts must match
: 06 13 02D9 673 : BEQL 5$ : If EQL OK
50 00' DO 02DB 674 : MOVL S^#SS$_BADPARAM,R0 : Set error code
: 00C9 31 02DE 675 : BRW 200$ : Take common exit
02E1 676 :
02E1 677 : Ignore event if EFI database is empty (no events get transmitted)
02E1 678 :
2000 8F 0A A8 B1 02E1 679 5$: : CMPW RAW$W_EVTCODE(R8),#EVC$C_VMS_DBC ; EFI database change
: 0A 13 02E7 680 : BEQL 10$ : If so, buffer regardless of EFI list
50 0000'CF DO 02E9 681 : MOVL NET$GL_CNR_EFI,R0 : Get address of EFI listhead
: 60 50 D1 02EE 682 : CMPL R0,(R0) : Is list empty?
: 3C 13 02F1 683 : BEQL 14$ : If so, exit ignoring the event
02F3 684 :
02F3 685 : If this is the first event to be buffered, then allocate an
02F3 686 : buffer to stored the event records until EVL picks them up.
02F3 687 :
: 0018'CF D5 02F3 688 10$: : TSTL EVT_L_BUFFER : Buffer allocated yet?
: 1A 12 02F7 689 : BNEQ 11$ : Branch if so
51 00001F4C 8F DO 02F9 690 : MOVL #12+NET$C_EVTBUFLTH,R1 : Set size of buffer needed
: FCFD' 30 0300 691 : BSBW NET$ALLOCATE : Allocate the buffer
: OD 50 E9 0303 692 : BLBC R0,11$ : If error, skip event reporting
: 0018'CF 0C A2 9E 0306 693 : MOVAB 12(R2),EVT_L_BUFFER : Store buffer pointer
001C'CF 0018'CF DO 030C 694 : MOVL EVT_L_BUFFER,EVT_L_BUFPTR ; Point to first available position
0313 695 :
0313 696 : If "lost event" already reported, allow 1 data base change event
0313 697 : to get thru
: 0313 698 :
: 01 E1 0313 699 11$: : BBC #EVT$V_LOSTEVENT,- : If BC then try to buffer event
: 20 000C'CF 0315 700 : EVT_B_FLAGS,20$ :
: 000F'CF 57 A0 0319 701 : ADDW R7,EVT_W_LOST ; Keep total of events lost

```

```

      OA A8 B1 031E 702 CMPW RAW$W EVTCODE(R8),- ; No space - see if database change
    2000 8F 0321 703 #EVC$C_VMS_DBC ;
      06 12 0324 704 BNEQ 12$ ; If NEQ no - ignore event
      02 E3 0326 705 BBCS #EVT$V_DBCEVENT,- ; If BC, database change not yet logged
    06 000C'CF 0328 706 EVT_B_FLAGS,15$ ;
      007C 30 032C 707 12$: BSBW STARTUP_EVL ; Start EVL process (if possible) in
      0075 31 032F 708 ; case it died and left our buffer full
    58 003E'CF 9E 0332 709 14$: BRW 100$ ; Nothing to do
      24 11 0337 710 15$: MOVAB DBC_EVENT,R8 ; Put in 'DBC event' event
      0339 711 BRB 25$ ; Log the database change
      0339 712 ;
      0339 713 ; If only room for one more event in buffer, insert "lost event"
      0339 714 ;
    50 001C'CF 00'8'CF C3 0339 715 20$: SUBL3 EVT_L_BUFFER,EVT_L_BUFPTR,R0 ; Compute # bytes in use
    50 00001F00 8F 50 C3 0341 716 SUBL3 R0,#NET$C_AVLBUFCTR,R0 ; Compute # bytes left
      50 57 B1 0349 717 CMPW R7,R0 ; Enough space for this event?
      12 1B 034C 718 BLEQU 30$ ; If LEQU yes
    000F'CF 57 A0 034E 719 ADDW R7,EVT_W_LOST ; Keep total of events lost
      02 88 0353 720 BISB2 #EVT$M_LOSTEVENT,- ; Show that an event has been lost
    58 000C'CF 9E 0355 721 EVT_B_FLAGS ;
      0020'CF 9E 0358 722 MOVAB LOST_EVENT,R8 ; Put in "lost event" event
      57 68 3C 035D 723 25$: MOVZWL (R8),R7 ; Get the length of the event
      0360 724 ;
      0360 725 ; Insert event into buffer
      0360 726 ;
    00000000'GF 7D 0360 727 30$: MOVQ G^EXE$GQ SYSTIME,- ; Time-stamp the event
      02 A8 0366 728 RAW$T SYSTIM(R8) ;
    001C'DF 68 57 28 0368 729 MOV3 R7,(R8),@EVT_L_BUFPTR ; Move event into the buffer
    001C'CF 53 D0 036E 730 MOVL R3,EVT_L_BUFPTR ; Update the pointer
      000D'CF B6 0373 731 INCW EVT_W_THRESH ; Another event in buffer
      0377 732 ;
      0377 733 ; If the event threshold has been reached, broadcast "events available" me
      0377 734 ;
    000D'CF B1 0377 735 CMPW #NET$C_EVTTHRESH,- ; Has the threshold been reached?
      05 0379 736 EVT_W_THRESH ;
      05 1E 037C 737 BGEQU 90$ ; If GEQU no
      01 88 037E 738 BISB2 #EVT$M_EVTAVL,- ; Set the flag
    000C'CF 0380 739 EVT_B_FLAGS ;
      0383 740 ;
      0383 741 ASSUME EVT$V_EVTAVL EQ 0 ;
      0383 742 ;
    1F 000C'CF E9 0383 743 90$: BLBC EVT_B_FLAGS,100$ ; If LBC can't send mbx msg yet
      0388 744 ;
      0388 745 ; It's OK to inform the world that the event buffer should be read
      0388 746 ;
      0388 747 ;
      0388 748 SEND_EVT_MSG: ;
      0388 749 ;
      0388 750 ; Startup EVL process if not already running
      0388 751 ;
      21 10 0388 752 BSBB STARTUP_EVL ; Startup EVL process if needed
      038A 753 ;
      038A 754 ; Reset the threshold timer
      038A 755 ;
      51 D4 038A 756 CLRL R1 ; Set up REQIDT for canceling timer
    52 FC70 CF 9E 038C 757 MOVAB EVT_TIMER,R2 ; Get action routine address for timer
    53 00000000 02FAF080 8F 7D 0391 758 MOVQ #NET$C_EVTTIMER,R3 ; Let this much time elapse

```

```
FC61' 30 039C 759      BSBW  WQESRESET_TIM      ; Cancel previous timer, set new one
        039F 760      :
        039F 761      : Now send the mailbox message
        039F 762      :
53  02  D0 039F 763      MOVL  #<1@MBX$V EVTAVL>,R3 ; Set mask
52  3E  3C 03A2 764      MOVZWL #MSG$ EVTAVL,R2    ; Set mbx msg code
        43  10 03A5 765      BSBB  BROADCAST          ; Broadcast the message
50  00'  3C 03A7 766 100$: MOVZWL S^#SS$ _NORMAL,R0 ; Indicate success
        05  03AA 767 200$: RSB
        03AB 768
        03AB 769      .DSABL LSB
```

```

03AB 771 .SBTTL STARTUP_EVL - Start EVL process
03AB 772 :+
03AB 773 : STARTUP_EVL - Start EVL process
03AB 774 :
03AB 775 : Start EVL process (if possible). This is done by queueing a WQE
03AB 776 : to do the job, since STARTUP_OBJ calls CNF action routines, some
03AB 777 : of which allocate the CNF static temporary buffer. Unfortunately,
03AB 778 : this is required because we may be logging counters while having
03AB 779 : the static temporary buffer allocated (specifically, CNT does this).
03AB 780 :
03AB 781 : Inputs:
03AB 782 :
03AB 783 : None
03AB 784 :
03AB 785 : Outputs:
03AB 786 :
03AB 787 : None
03AB 788 :
03AB 789 : R0 destroyed.
03AB 790 :-
03AB 791 :
03AB 792 STARTUP_EVL:
FC52' 30 03AB 793 BSBW WQES$FORK ; Fork to work queue level
52 7C 03AE 794 CLRQ R2 ; Pass nothing as SYS$NET to EVL
54 7C 03B0 795 CLRQ R4 ; Use default process name
58 1A 9A 03B2 796 MOVZBL #EVL_OBJ,R8 ; Object number of EVL
FC48' 30 03B5 797 BSBW NET$STARTUP_OBJ ; Create EVL process
03B8 798 ; ....ignore any errors
05 03B8 799 RSB

```

```

03B9 801 .SBTTL Event logging database changes
03B9 802 :+
03B9 803 : NET$DBC_ESI - note the receiver database changed
03B9 804 : NET$DBC_EFI - note the xmitter database changed
03B9 805 :
03B9 806 : INPUTS: NONE
03B9 807 :
03B9 808 : OUTPUTS: R0 Low bit set
03B9 809 :
03B9 810 : All other registers are preserved
03B9 811 :
03B9 812 :-
03B9 813 NET$DBC_EFI::
58 OFFE 8F BB 03B9 814 PUSHB #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
EC 10 03BD 815 BSBB STARTUP EVL ; Startup EVL if needed
003E'CF 9E 03BF 816 MOVAB DBC_EVENT,R8 ; Point to event buffer
57 68 3C 03C4 817 MOVZWL (R8),R7 ; Get length of item
FFOC 30 03C7 818 BSBW INTERNAL_EVENT ; Inform EVL of EFI database change
52 0044 8F 3C 03CA 819 MOVZWL #MSG$ EVT_XMTCHG,R2 ; This is the mailbox message code
53 08 D0 03CF 820 MOVL #<1@MBX$V EVT_XMTCHG>,R3 ; Set mask
OC 11 03D2 821 BRB DBC_COMMON ; Finish in common code
03D4 822
03D4 823 NET$DBC_ESI::
OFFE 8F BB 03D4 824 PUSHB #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
D1 10 03D8 825 BSBB STARTUP EVL ; Startup EVL if needed
52 3F 3C 03DA 826 MOVZWL #MSG$ EVTRCVCHG,R2 ; This is the mailbox message code
53 04 D0 03DD 827 MOVL #<1@MBX$V_EVTRCVCHG>,R3 ; Set mask
03E0 828
03E0 829 DBC_COMMON:
OFFE 08 10 03E0 830 BSBB BROADCAST ; Broadcast the message
8F BA 03E2 831 POPR #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
50 01 D0 03E6 832 MOVL #1,R0 ; Always successful
05 03E9 833 RSB ; Done
03EA 834 :+
03EA 835 :
03EA 836 : BROADCAST - broadcast event related message
03EA 837 :
03EA 838 : INPUTS:
03EA 839 : R2 - MSG$ code
03EA 840 : R3 - mask bit for mailbox filtering !*** not yet used
03EA 841 :
03EA 842 :-
55 0000'CF D0 03EA 843 BROADCAST:
54 D4 03EF 844 MOVL NET$GL_PTR_UCB0,R5 ; Point to a NET CB address
50 0A D0 03F1 845 CLRL R4 ; No message text
FC09' 30 03F4 846 MOVL #NETUP$ BRDCST,R0 ; Function is "broadcast"
05 03F7 847 BSBW CALL_NETDRIVER ; Call driver com routine
848 RSB ; Done

```

```

03F8 850 .SBTTL Outbound raw event processing
03F8 851 :
03F8 852 : NET$READ_EVENT - Read out event buffer
03F8 853 :
03F8 854 : FUNCTIONAL DESCRIPTION:
03F8 855 :
03F8 856 : This routine places the event buffer into the specified result (P4)
03F8 857 : buffer. Flags, pointers, and thresholds are reset for more logging.
03F8 858 :
03F8 859 : INPUTS: NET$GL_PTR_P3 - Address of result length word
03F8 860 : NET$GL_PTR_P4 - Address of result buffer
03F8 861 : NET$GL_SIZ_P4 - Size of result buffer
03F8 862 :
03F8 863 : OUTPUTS: P3, P4 have length of result buffer and result buffer
03F8 864 : R0 - low word has status (NORMAL); high word has byte count
03F8 865 :
03F8 866 :
03F8 867 NET$READ_EVENT::
50 0018'CF C3 03F8 868 SUBL3 EVT_L_BUFFER,- ; Entry
001C'CF 03FC 869 ; Compute no. of bytes in buffer
56 50 DO 0400 870 MOVL R0,R6 ; Save bfr lth
57 0000'CF DO 0403 871 MOVL NET$GL_SIZ_P4,R7 ; Get size of result buffer
56 57 D1 0408 872 CML R7,R6 ; Is there room in the result bfr?
1C 1E 040B 873 BGEQU 10$ ; If GEQU yes
040D 874 :
040D 875 : There's not enough room in the caller's buffer to hold
040D 876 : all the events we have buffered. Scan our event buffer
040D 877 : and find the last event that will fit, so that we always
040D 878 : copy "whole" event records.
040D 879 :
51 0018'CF DO 040D 880 MOVL EVT_L_BUFFER,R1 ; Get buffer pointer
52 57 DO 0412 881 MOVL R7,R2 ; Get length of user's buffer
53 61 3C 0415 882 5$: MOVZWL (R1),R3 ; Get length of next event
52 53 D1 0418 883 ; Will next event fit in buffer?
08 1A 041B 884 BGRU 8$ ; If not, then stop here
52 53 C2 041D 885 ; If it fits, then include it
51 53 C0 0420 886 ADDL R3,R1 ; Skip to next event
50 57 F0 11 0423 887 BRB 5$ ; Keep scanning
57 52 C3 0425 888 8$: SUBL3 R2,R7,R0 ; Compute size of bytes to move
0429 889 :
0429 890 : The number of bytes to be moved has been determined.
0429 891 : Store the byte count in the P3 result length word.
0429 892 :
51 0000'CF DO 0429 893 10$: MOVL NET$GL_PTR_P3,R1 ; Get address of result length
03 13 042E 894 BEQL 20$ ; If EQL there is none
61 50 B0 0430 895 MOVW R0,(R1) ; Store the result length
0433 896 :
0433 897 : Construct the final IOSB with the byte count
0433 898 :
7E 50 B0 0433 899 20$: MOVW R0,-(SP) ; Byte count to high word
7E 00' B0 0436 900 MOVW S^#SS$_NORMAL,-(SP) ; Store I/O status in low word
0439 901 :
0439 902 : Move the events into the caller's buffer
0439 903 :
0018'DF 02 AE 28 0439 904 MOVCL 2(SP),@EVT_L_BUFFER,- ; Move event buffer to result bfr
0000'DF 043F 905 @NET$GL_PTR_P4
0442 906 :

```

				0442	907	:	Shift down any remaining events that couldn't be copied					
				0442	908	:	to the front of the buffer.					
				0442	909	:						
	52	02	AE	3C	0442	910	MOVZWL	2(SP),R2	:	Get the number of bytes we moved		
	50	56	52	C3	0446	911	SUBL3	R2,R6,R0	:	Compute # bytes of remaining events		
	001C	'CF	52	C2	044A	912	SUBL	R2,EVT_L_BUFPTR	:	Adjust buffer pointer		
0018	'DF	61	50	28	044F	913	MOV3	R0,(R1),EVT_L_BUFFER	:	Move remaining evts to bfr top		
0011	'CF	000F	'CF	B1	0455	914	CMPW	EVT_W_LOST,EVT_W_PEAK	:	Did we hit peak "lost bytes"?		
			07	1B	045C	915	BLEQU	30\$	:	Branch if not		
0011	'CF	000F	'CF	B0	045E	916	MOVW	EVT_W_LOST,EVT_W_PEAK	:	Store new peak "lost bytes"		
		000F	'CF	B4	0465	917	CLRW	EVT_W_LOST	:	Clear lost count statistic		
					0469	918	CLRBIT	#EVT\$V_LOSTEVENT,-	:	There's now room in the buffer		
					0469	919		EVT_B_FLAGS	:			
	56	02	AE	B1	046F	920	CMPW	2(SP),R6	:	Did we empty the buffer?		
			08	13	0473	921	BEQL	50\$	:	If so, then indicate buffer empty		
					0475	922	SETBIT	#EVT\$V_EVTAVL,EVT_B_FLAGS	:	Tell EVL to read more events		
			08	11	047B	923	BRB	100\$	:	proceed		
			000C	'CF	94	047D	924	50\$:	CLRB	EVT_B_FLAGS	:	Reset the flags
			000D	'CF	B4	0481	925	100\$:	CLRW	EVT_W_THRESH	:	Reset the event threshold
			50	8E	D0	0485	926	100\$:	POPL	R0	:	Get 1st IOSB longword
				05	0488	927	RSB		:	Done		



```
0489 929 .SBTTL NET$SET_CTR_TIMER - Reset automatic counter timer
0489 930 :+
0489 931 : NET$SET_CTR_TIMER - Reset automatic counter timer
0489 932 :
0489 933 : FUNCTIONAL DESCRIPTION
0489 934 :
0489 935 : This routine is called whenever the a data base is updated to start or
0489 936 : reset the automatic counter timer. When the counter timer fires, the
0489 937 : counters will be logged on whatever CNFs are due. The timer is then
0489 938 : reset to the next earliest due time.
0489 939 :
0489 940 : Inputs:
0489 941 :
0489 942 : R11 = CNR address
0489 943 : R10 = CNF address
0489 944 :
0489 945 : Outputs:
0489 946 :
0489 947 : None
0489 948 :
0489 949 : R0-R9 are destroyed.
0489 950 :-
0489 951 :-
0489 952 NET$SET_CTR_TIMER:: : Reset logging counter timer
56 0018'CF 0F 0489 953 -MOVAB CNX_CRI,R6 : Assume CRI data base
0000'CF 5B 048E 954 -Cmpl R11,NET$GL_CNR_CRI : Is it?
13 0493 955 -BEQL 10$ : If EQL then yes
56 0030'CF 19 13 0493 955 -BEQL 10$ : If EQL then yes
0000'CF 5B D1 049A 957 -Cmpl R11,NET$GL_CNR_NDI : Assume NDI data base
0D 049F 958 -BEQL 10$ : Is it the NDI data base
56 0000'CF 0D 9E 04A1 959 -MOVAB CNX_PLI,R6 : If EQL then yes
0000'CF 5B D1 04A6 960 -Cmpl R11,NET$GL_CNR_PLI : Assume PLI data base
01 13 04AB 961 -BEQL 10$ : Is it?
05 05 04AD 962 -RSB : Branch if so
04AE 963 : Else, unsupported database
04AE 964 :
04AE 965 : Since it is common for many CNF blocks to be updated by the
04AE 966 : network manager at the same time, it is possible to reduce the
04AE 967 : total amount of work to be done somewhat by waiting a short time,
04AE 968 : the so called "suppression interval", before running the timer
04AE 969 : update algorithm after any given CNF block is updated. This has
04AE 970 : the effect of batching the requests and reduces the work by making
04AE 971 : better use of each scan of the data base.
04AE 972 :
04AE 973 : The suppression timer interval is 2 seconds. This is long enough
04AE 974 : for a typical NCP>SET KNOWN NODES ALL command to complete, and
04AE 975 : short enough not to be noticed by the issuer of the command.
04AE 976 :
130$ 04AE 977 10$: -MOVL CNX$SL DEL TIME(R6),R9 : Get the counter timer field i.d.
59 08 A6 DO 04AE 977 10$: -BSBW CNF$GET_FIELD : Get its value
FB4B' 30 04B2 978 -BLBC R0,15$ : If LBC then its not set
1D 50 E9 04B5 979 -ADDL G^EXE$GL ABSTIM,R8 : Convert to absolute time
58 00000000'GF C0 04B8 980 -MOVL CNX$SL ABS TIME(R6),R9 : Get field i.d.
59 0C A6 DO 04BF 981 -BSBW CNF$POT_FIELD : Store it
FB3A' 30 04C3 982 -MOVZBL CNX$B TIM SUP(R6),R0 : Get the suppression timer bit no.
50 01 A6 9A 04C6 983 -BBSS R0,EVT_B_FLAGS,15$ : If BS then update suppression timer
05 000C'CF 50 E2 04CA 984 : is ticking
04D0 985
```

```

58 02 D0 04D0 986      MOVL #2,R8      : Suppress processing request for 2 sec
    71 11 04D3 987      BRB 40$        : Set the timer
0086 31 04D5 988 15$:  BRW 50$        : Continue
    04D8 989
    04D8 990
    04D8 991 20$:  :
    04D8 992      : Entry point called when timer fires.
    04D8 993      :
    04D8 994      : Determine database
    04D8 995      :
55 51 50 55 D0 04D8 996      MOVL R5,R0      : Get the timer WQE for deallocation
    10 10 EF 04DB 997      EXTZV #16,#16,R1,R5 : Get timer database i.d.
    FB1D' 30 04E0 998      BSBW NET$DEALLOCATE : Deallocate WQE
56 0018'CF 9E 04E3 999      MOVAB CNX_CRI,R6   : Assume CRI timer
    03 55 B1 04E8 1000      CMPW R5,#EVC$C_SRC_CIR : Is it?
    18 13 04EB 1001      BEQL 25$        : If EQL yes
56 0030'CF 9E 04ED 1002      MOVAB CNX_NDI,R6   : Assume NDI timer
    00 55 B1 04F2 1003      CMPW R5,#EVC$C_SRC_NOD : Is it?
    OE 13 04F5 1004      BEQL 25$        : If EQL yes
56 0000'CF 9E 04F7 1005      MOVAB CNX_PLI,R6   : Assume PLI timer
    01 55 B1 04FC 1006      CMPW R5,#EVC$C_SRC_LIN : Is it?
    04 13 04FF 1007      BEQL 25$        : Branch if so
    0501 1008
    0501 1009      BUG_CHECK NETNOSTATE,FATAL : Timer i.d. unknown
    0505 1010
58 14 B6 D0 0505 1011 25$:  MOVL @CNX$C_CNR_PTR(R6),R11 : Get the CNR pointer
50 01 A6 9A 0509 1012      MOVZBL CNX$B_TIM_SUP(R6),R0 : Get the suppression timer bit no.
    050D 1013      CLRBIT R0,EVT_B_FLAGS : Suppression timer no longer ticking
    4A 10 0513 1014      BSBW TICK        : Process CNF timers
    0515 1015      :
    0515 1016      : Determine the next earliest CNF due time
    0515 1017      :
59 5A D4 0515 1018      CLRL R10        : Start from the head of the CNF list
    OC A6 D0 0517 1019      MOVL CNX$C_ABS_TIME(R6),R9 : Get absolute time field i.d.
    51 04 D0 051B 1020      MOVL #NFB$C_OP_FNDMIN,R1 : Fct is "find minimum value"
    FADF' 30 051E 1021      BSBW CNF$KEY_SEARCH : Find minimum value
    3A 50 E9 0521 1022      BLBC R0,50$     : If no CNF found, no timers are set
    FAD9' 30 0524 1023      BSBW CNF$GET_FIELD : Get due time of minimum CNF
    34 50 E9 0527 1024      BLBC R0,50$     : Branch if cannot get it
58 10 B6 58 D0 052A 1025      MOVL R8,@CNX$C_OLD_TIME(R6) : Store the absolute due time
    00000000'GF D1 052E 1026      CMPL G^EXE$GL_ABSTIM,R8 : Have we passed that time yet?
    0535 1027      : (this could happen if the event
    0535 1028      : buffer is full)
    0535 1029      : If LSSU then no
    58 05 1F 0535 1029      BLSSU 35$      :
    02 D0 0537 1030      MOVL #2,R8      : Try again in 2 seconds
    0A 11 053A 1031      BRB 40$        : Continue
58 00000000'GF C2 053C 1032 35$:  SUBL G^EXE$GL_ABSTIM,R8 : Convert to delta time
    58 02 C0 0543 1033      ADDL #2,R8      : CNF timers are grouped into 2 second
    0546 1034      : buckets to batch the work
    0546 1035 40$:  :
    0546 1036      : Reset the timer
    0546 1037      :
53 00 00989680 8F 58 7A 0546 1038      EMUL R8,#10*1000*1000,#0,R3 : Get quadword timer interval
    52 86 AF 9E 054F 1039      MOVAB 20$,R2     : Setup timer routine address
    51 66 D0 0553 1040      MOVL CNX$W_ID_CTM-2(R6),R1 : Setup timer i.d. in high order word
    51 0300 8F B0 0556 1041      MOVW #WQE$C_QUAL_CTM@8,R1 : Setup timer qualifier
    FAA2' 30 055B 1042      BSBW WQE$RESET_TIM : Reset the counter timer

```

```

05 055E 1043 50$: RSB
055F 1044
055F 1045
055F 1046
055F 1047
00000000'GF D0 055F 1048 TICK: MOVL G^EXE$GL_ABSTIM,- ; Get seconds since boot to be used
0013'CF 0565 1049 ; BASE_TIME ; as the common base for updating timers
5A D4 0568 1050 CLRL R10 ; Start from the head of the CNF list
056A 1051 10$:
056A 1052 ;
056A 1053 ; Find the next CNF whose timer is due. Must first check for
056A 1054 ; CNF entries whose time is past due to prevent finding the same
056A 1055 ; CNFs over and over again when there are more entries than can fit
056A 1056 ; in the event buffer.
50 001C'CF 0018'CF C3 056A 1057 SUBL3 EVT_L_BUFFER,EVT_L_BUFPTR,R0 ; Compute # bytes in use
50 00001F00 8F 50 C3 0572 1058 SUBL3 RO,#NET$C_AVLBUFCTR,R0 ; Compute # bytes left
50 0064 8F B1 057A 1059 CMPW #100,R0 ; Enough room in buffer?
59 0C A6 6E 1A 057F 1060 BGTRU 40$ ; If GTRU then no
58 10 B6 04 D0 0581 1061 MOVL CNX$ABS_TIME(R6),R9 ; Get field i.d.
00000000'GF 58 D1 0585 1062 ADDL3 #4,@CNX$OLD_TIME(R6),R8 ; Get due time of oldest CNFs
0B 1A 058A 1063 CMPL R8,G^EXE$GL_ABSTIM ; Use 4 second interval but don't
51 01 D0 0591 1064 BGTRU 13$ ; exceed the current time
FA67' 30 0593 1065 MOVL S^#NFB$C_OP_GTRU,R1 ; Match on key value GTRU CNF field
13 50 E8 0596 1066 BSBW CNF$KEY_SEARCH ; Find Appropriate CNF
5A D4 0599 1067 BLBS RO,15$ ; If LBS then found one
58 00000000'GF 01 C1 059C 1068 CLRL R10 ; Start next scan from head of CNF list
059E 1069 13$: ADDL3 #1,G^EXE$GL_ABSTIM,R8 ; Bias current time. The '+1' is used
05A6 1070 ; to help smooth the coarseness of the
05A6 1071 ; timer and to amortize the timer over-
05A6 1072 ; head across a number of CNFs.
51 01 D0 05A6 1073 MOVL S^#NFB$C_OP_GTRU,R1 ; Match on key value GTRU CNF field
FA54' 30 05A9 1074 BSBW CNF$KEY_SEARCH ; Find Appropriate CNF
40 50 E9 05AC 1075 BLBC RO,40$ ; If LBC then no such CNF
50 0000'DF 0F 05AF 1076 15$: REMQUE @NET$GQ_TMP_BUF,R0 ; Drain temp buffer queue since
05 1D 05B4 1077 ; the search below could fill it
FA47' 30 05B4 1078 BVS 17$ ; If VS then none
F4 11 05B6 1079 BSBW NET$DEALLOCATE ; Deallocate the block
05B8 1080 BRB 15$ ; Loop
05BB 1081 17$:
05BB 1082 ;
05BB 1083 ; Snapshot the counters and log the event records. The CNT
05BB 1084 ; action routine will log the event record because the CLRCNT
05BB 1085 ; flag is set.
0000'CF DD 05BB 1086 PUSHL NET$GL_FLAGS ; Save current flags
59 04 A6 D0 05BF 1087 SETBIT #NET$V_CLRCNT,NET$GL_FLAGS ; Counters are to be cleared
FA34' 30 05C5 1088 MOVL CNX$COUNTER(R6),R9 ; Get counter field i.d.
0000'CF 8ED0 05C9 1089 BSBW CNF$GET_FIELD ; Read/clear the counters
05CC 1090 POPL NET$GL_FLAGS ; Restore flags
05D1 1091 ;
05D1 1092 ; Calculate its next due time and store it in the CNF
05D1 1093 ;
59 08 A6 D0 05D1 1094 MOVL CNX$DEL_TIME(R6),R9 ; Get delta time field i.d.
FA28' 30 05D5 1095 BSBW CNF$GET_FIELD ; Fetch it
59 0C A6 D0 05D8 1096 MOVL CNX$ABS_TIME(R6),R9 ; Get absolute time field i.d.
05 50 E8 05DC 1097 BLBS RO,20$ ; If LBS then delta time was valid
FA1E' 30 05DF 1098 BSBW CNF$CLR_FIELD ; Else the timer has been cancelled
08 11 05E2 1099 BRB 30$ ; Continue

```

NETEVTLOG  
V04-000

M 10  
- Process Event logging needs 16-SEP-1984 01:25:34 VAX/VMS Macro V04-00 Page 25  
NET\$SET\_CTR\_TIMER - Reset automatic coun 5-SEP-1984 02:20:54 [NETACP.SRC]NETEVTLOG.MAR;1 (11)

```
58 0013'CF  C0 05E4 1100 20$: ADDL BASE_TIME_R8 ; Determine when timer should fire next
    FA14'  30 05E9 1101      BSBW CNF$PUT_FIELD ; Store it
    FF7B  31 05EC 1102 30$: BRW 10$ ; Loop
    05 05EF 1103 40$: RSB
      05FO 1104
      05FO 1105
      05FO 1106 .END
```

NETEVTLOG  
Symbol table

- Process Event logging needs

N 10

16-SEP-1984 01:25:34 VAX/VMS Macro V04-00  
5-SEP-1984 02:20:54 [NETACP.SRC]NETEVTLOG.MAR;1

Page 26  
(11)

ABS	0000014B	R	04	EVCSC_DLL_LSC	= 00000140
ACH	0000016A	R	04	EVCSC_DLL_PNEW	= 00000001
ACPSC_STA_F	= 00000004			EVCSC_DLL_POLD	= 00000000
ACPSC_STA_H	= 00000005			EVCSC_DLL_RSC	= 00000141
ACPSC_STA_I	= 00000000			EVCSC_NMA_ABS	= 00000007
ACPSC_STA_N	= 00000001			EVCSC_NMA_CTR	= 00000008
ACPSC_STA_R	= 00000002			EVCSC_NMA_LOS	= 00000000
ACPSC_STA_S	= 00000003			EVCSC_NMA_PRSN	= 00000003
ADJ\$W_PNA	= 00000004			EVCSC_NMA_ZER	= 00000009
BASE_TIME	00000013	R	02	EVCSC_NSL_DBR	= 000000C2
BIT...	= 00000006			EVCSC_SCL_LNS	= 00000080
BROADCAST	000003EA	R	04	EVCSC_SCL_PNEW	= 00000002
BUG\$NETNOSTATE	*****	X	04	EVCSC_SCL_POLD	= 00000001
CALL_NETDRIVER	*****	X	04	EVCSC_SCL_PRSN	= 00000000
CD1	00000170	R	04	EVCSC_SRC_ARE	= 00000005
CD1_2	00000192	R	04	EVCSC_SRC_CIR	= 00000003
CIR_ADJ	00000148	R	04	EVCSC_SRC_LIN	= 00000001
CIR_BEG	000000E4	R	04	EVCSC_SRC_NOD	= 00000000
CIR_COU	000000B8	R	04	EVCSC_SRC_NON	= 000000FF
CIR_PKT	000000D8	R	04	EVCSC_TPL_ACH	= 00000111
CIR_REASON	00000151	R	04	EVCSC_TPL_APL	= 00000100
CNFSCLR_FIELD	*****	X	04	EVCSC_TPL_ARJ	= 00000110
CNFSGET_FIELD	*****	X	04	EVCSC_TPL_AUP	= 0000010F
CNFSKEY_SEARCH	*****	X	04	EVCSC_TPL_ILF	= 0000010B
CNFSPUT_FIELD	*****	X	04	EVCSC_TPL_IOF	= 0000010D
CNFS_ADVANCE	= 00000000			EVCSC_TPL_ISF	= 0000010C
CNFS_QUIT	= 00000002			EVCSC_TPL_LDF	= 00000107
CNFS_TAKE_CURR	= 00000003			EVCSC_TPL_LDO	= 00000113
CNFS_TAKE_PREV	= 00000001			EVCSC_TPL_LDS	= 00000112
CNX\$B_SPARE	= 00J00000			EVCSC_TPL_LUP	= 0000010A
CNX\$B_TIM_SUP	= 00000001			EVCSC_TPL_OPL	= 00000103
CNX\$C_LENGTH	= 00000018			EVCSC_TPL_PADJ	= 00000008
CNX\$C_ABS_TIME	= 0000000C			EVCSC_TPL_PFM	= 00000104
CNX\$C_CNR_PTR	= 00000014			EVCSC_TPL_PHIA	= 00000002
CNX\$C_COUNTER	= 00000004			EVCSC_TPL_PNOD	= 00000003
CNX\$C_DEL_TIME	= 00000008			EVCSC_TPL_PPKB	= 00000001
CNX\$C_OLD_TIME	= 00000010			EVCSC_TPL_PPKH	= 00000000
CNX\$W_ID_CTM	= 00000002			EVCSC_TPL_PRSN	= 00000005
CNX_CRI	00000018	R	03	EVCSC_TPL_PRU	= 00000105
CNX_CRI_OLDTIM	00000004	R	02	EVCSC_TPL_PSTS	= 00000007
CNX_NDI	00000030	R	03	EVCSC_TPL_PVRS	= 00000006
CNX_NDI_OLDTIM	00000008	R	02	EVCSC_TPL_RCH	= 0000010E
CNX_PLI	00000000	R	03	EVCSC_TPL_RPL	= 00000102
CNX_PLI_OLDTIM	00000000	R	02	EVCSC_TPL_UPL	= 00000101
COU	000000C0	R	04	EVCSC_TPL_VFR	= 00000106
COUNTER	000000A0	R	04	EVCSC_VMS_DBC	= 00002000
DBC_COMMON	000003E0	R	04	EVL_OBJ	= 0000001A
DBC_EVENT	0000003E	R	02	EVT\$M_CST_CRI	= 00000010
ENTER_NO_SRC	000001B3	R	04	EVT\$M_CST_NDI	= 00000020
ENTER_PKTHDR	00000220	R	04	EVT\$M_CST_PLI	= 00000008
ENTER_PPKB	00000249	R	04	EVT\$M_DBCEVENT	= 00000004
ENTER_SRCAREA	000001B8	R	04	EVT\$M_EVTAVL	= 00000001
ENTER_SRCCIR	000001F5	R	04	EVT\$M_LOSTEVENT	= 00000002
ENTER_SRCLIN	00000214	R	04	EVT\$S_CST_CRI	= 00000001
ENTER_SRCNOD	000001D1	R	04	EVT\$S_CST_NDI	= 00000001
ENT_17	00000202	R	04	EVT\$S_CST_PLI	= 00000001
ENT_SRC	000001FF	R	04	EVT\$S_DBCEVENT	= 00000001

NETEVTLOG  
Symbol table

- Process Event logging needs

B 11

16-SEP-1984 01:25:34 VAX/VMS Macro V04-00  
5-SEP-1984 02:20:54 [NETACP.SRC]NETEVTLOG.MAR;1

Page 27  
(11)

NET  
V04

```

EVTSS_EVTAVL           = 00000001
EVTSS_LOSTEVENT       = 00000001
EVTSSV_CST_CRI        = 00000004
EVTSSV_CST_NDI        = 00000005
EVTSSV_CST_PLI        = 00000003
EVTSSV_DBCEVENT       = 00000002
EVTSSV_EVTAVL         = 00000000
EVTSSV_LOSTEVENT     = 00000001
EVT_B_FLAGS           = 0000000C R      02
EVT_L_BUFFER          = 00000018 R R    02
EVT_L_BUFPTR          = 0000001C R R    02
EVT_TIMER              = 00000000 R R    04
EVT_W_LOST            = 0000000F R R    02
EVT_W_PEAK            = 00000011 R R    02
EVT_W_THRESH          = 0000000D R      02
EXESGC_ABSTIM         = ***** X      04
EXESGQ_SYSTIME        = ***** X      04
GET_NDI                = 000002A5 R      04
ILF                    = 00000151 R R    04
INTERNAL_EVENT        = 000002D6 R R    04
IOF                    = 00000113 R R    04
ISF                    = 00000138 R R    04
LDF                    = 00000151 R R    04
LDO                    = 0000013D R R    04
LDS                    = 0000013D R R    04
LIN_COU                = 000000BD R R    04
LNS                    = 0000017C R R    04
LOST_EVENT            = 00000020 R R    02
LSC                    = 0000019E R      04
MBXSV_EVTAVL          = 00000001
MBXSV_EVTRCVCHG       = 00000002
MBXSV_EVTXMTCHG       = 00000003
MSGSV_EVTAVL          = 0000003E
MSGSV_EVTRCVCHG       = 0000003F
MSGSV_EVTXMTCHG       = 00000044
NET$AB_EVT_WQE        = 0000005C RG    02
NET$ALLOCATE          = ***** X      04
NET$C_ACT_TIMER       = 0000001E
NET$C_AVLBUFLTH       = 00001F00
NET$C_EFN_ASYN        = 00000002
NET$C_EFN_WAIT        = 00000001
NET$C_EVTBUFLTH       = 00001F40
NET$C_EVTTHRESH       = 00000005
NET$C_EVTTIMER        = 02FAF080
NET$C_IPL              = 00000008
NET$C_LSTEVTLTH       = 00000020
NET$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$C_MAX_NCB         = 00C0006E
NET$C_MAX_NODES       = 000003FF
NET$C_MAX_OBJ         = 000000FF
NET$C_MAX_WQE         = 00000014

```

```

NET$C_MINBUFSIZ       = 000000C0
NET$C_TID_ACT         = 00000003
NET$C_TID_RUS         = 00000001
NET$C_TID_XRT         = 00000002
NET$C_TRCTL_CEL       = 00000002
NET$C_TRCTL_OVR       = 00000005
NET$C_UTLBUFSIZ       = 00001000
NET$DBC_EFI           = 000003B9 RG    04
NET$DBC_ESI           = 000003D4 RG    04
NET$DEALLOCATE        = ***** X      04
NET$EVT_INTRAW        = 00000017 RG    04
NET$FIND_ADJ          = ***** X      04
NET$GETUTLBUF         = ***** X      04
NET$GL_CNR_CRI        = ***** X      03
NET$GL_CNR_EFI        = ***** X      04
NET$GL_CNR_NDI        = ***** X      03
NET$GL_CNR_PLI        = ***** X      03
NET$GL_FLAGS          = ***** X      04
NET$GL_INITVER        = ***** X      04
NET$GL_PTR_P2         = ***** X      04
NET$GL_PTR_P3         = ***** X      04
NET$GL_PTR_P4         = ***** X      04
NET$GL_PTR_UCBO       = ***** X      04
NET$GL_PTR_VCB        = ***** X      04
NET$GL_SIZ_P2         = ***** X      04
NET$GL_SIZ_P4         = ***** X      04
NET$GL_UTLBUF         = ***** X      04
NET$GQ_TMP_BUF        = ***** X      04
NET$LOG_EVENT         = 000002CC RG    04
NET$M_MAXLNKMSK      = 000003FF
NET$NDI_BY_ADD        = ***** X      04
NET$READ_EVENT        = 000003F8 RG    04
NET$SET_CTR_TIMER     = 00000480 RG    04
NET$STARTUP_OBJ       = ***** X      04
NET$V_CLRNT           = 00000002
NETUPDS_BRDCST       = 0000000A
NFBSC_CRI_CNT         = 04020044
NFBSC_CRI_CTA         = 04010011
NFBSC_CRI_LCT         = 04010015
NFBSC_CRI_NAM         = 04020041
NFBSC_NDI_CNT         = 02020042
NFBSC_NDI_CTA         = 02010011
NFBSC_NDI_CTI         = 02010013
NFBSC_NDI_NNA        = 02020043
NFBSC_OP_FNDMIN      = 00000004
NFBSC_OP_GTRU        = 00000001
NFBSC_PLI_CNT        = 05020044
NFBSC_PLI_CTA        = 05010010
NFBSC_PLI_LCT        = 05010013
NFBSC_PLI_NAM        = 05020041
NMACS_PTY_AI         = 00000040
NMACS_PTY_CD1        = 00000081
NMACS_PTY_CM1        = 000000C1
NMACS_PTY_CM2        = 000000C2
NMACS_PTY_CM3        = 000000C3
NMACS_PTY_CM4        = 000000C4
NMACS_PTY_DU1        = 00000001

```

NETEVTLOG  
Symbol table

- Process Event logging needs

C 11

16-SEP-1984 01:25:34  
5-SEP-1984 02:20:54

VAX/VMS Macro V04-00  
[NETACP.SRC]NETEVTLOG.MAR;1

Page 28  
(11)

NETI  
V04.

NMASC_PTY_DU2	=	00000002		
NMASC_PTY_H1	=	00000021		
NMASC_PTY_HI	=	00000020		
NOD_COU		000000B3	R	04
NON_PKT		000000D2	R	04
NSPSC_EXT_LNK	=	0000001E		
NSPSC_MAXHDR	=	00000009		
PNA_NODE		00000262	R	04
PRU		000000F0	R	04
RAWSB_SRCTYP		0000000C		
RAWSC_SIZE		0000001F		
RAWK_SIZE		0000001F		
RAWST_DATA		0000001E		
RAWST_SRCID		0000000D		
RAWST_SYSTM		00000002		
RAWSW_BYTES		00000000		
RAWSW_EVTCODE		0000000A		
RCBSW_ADDR	=	0000000E		
RCH		00000162	R	04
RSC		0000019E	R	04
SEND_EVT_MSG		00000388	R	04
SIZ...	=	00000001		
SS\$BADPARAM		*****	X	04
SS\$NORMAL		*****	X	04
STARTUP_EVL		000003AB	R	04
SUPPRESS_AREA		*****	X	04
TICK		0000055F	R	04
TRSC_MAXHDR	=	0000001C		
TRSC_NI_ALLEND1	=	040000AB		
TRSC_NI_ALLEND2	=	00000000		
TRSC_NI_ALLROU1	=	030000AB		
TRSC_NI_ALLROU2	=	00000000		
TRSC_NI_PREFIX	=	000400AA		
TRSC_NI_PROT	=	00000360		
TRSC_PRT_ECL	=	0000001F		
TRSC_PRI_RTHRU	=	0000001F		
VFR		0000010A	R	04
WQESB_EVL_DT1	=	0000001E		
WQESB_EVL_DT2	=	0000001F		
WQESC_LENGTH	=	00000024		
WQESC_QUAL_CTM	=	00000003		
WQESDEALLOCATE		*****	X	04
WQESFORK		*****	X	04
WQESL_EVL_PKT	=	00000018		
WQESRESET_TIM		*****	X	04
WQESW_ADJ_INX	=	00000020		
WQESW_EVL_CODE	=	0000001C		
WQESW_REQIDT	=	00000012		
_SS_	=	00000000		

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

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	0000001F ( 31.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
NET_IMPURE	00000080 ( 128.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
NET_PURE	00000048 ( 72.)	03 ( 3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
NET_CODE	000005F0 ( 1520.)	04 ( 4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.07	00:00:00.36
Command processing	155	00:00:01.01	00:00:04.50
Pass 1	601	00:00:23.83	00:00:32.66
Symbol table sort	0	00:00:03.08	00:00:03.16
Pass 2	257	00:00:04.82	00:00:06.17
Symbol table output	36	00:00:00.26	00:00:00.26
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1084	00:00:33.10	00:00:47.14

The working set limit was 1950 pages.  
126517 bytes (248 pages) of virtual memory were used to buffer the intermediate code.  
There were 130 pages of symbol table space allocated to hold 2195 non-local and 76 local symbols.  
1106 source lines were read in Pass 1, producing 25 object records in Pass 2.  
39 pages of virtual memory were used to define 35 macros.

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

Macro library name	Macros defined
-\$255\$DUA28:[SHRLIB]NMALIBRY.MLB;1	1
-\$255\$DUA28:[SHRLIB]EVCDEF.MLB;1	2
-\$255\$DUA28:[NETACP.OBJ]NETDRV.MLB;1	0
-\$255\$DUA28:[NETACP.OBJ]NET.MLB;1	14
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	1
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	26

2325 GETS were required to define 26 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:NETEVTLOG/OBJ=OBJ\$:NETEVTLOG MSRC\$:NETEVTLOG/UPDATE=(ENH\$:NETEVTLOG)+EXECMLS/LIB+LIB\$:NET/LIB+LIB\$:NETDRV/LIB+SHRLIB\$



