```
EEEEEEEEEEEEEEE  VVV         VVV  LLL
EEEEEEEEEEEEEEE  VVV         VVV  LLL
EEEEEEEEEEEEEEE  VVV         VVV  LLL
EEE              VVV         VVV  LLL
EEE              VVV         VVV  LLL
EEE              VVV         VVV  LLL
EEE              VVV         VVV  LLL
EEE              VVV         VVV  LLL
EEE              VVV         VVV  LLL
EEEEEEEEEEEE     VVV         VVV  LLL
EEEEEEEEEEEE     VVV         VVV  LLL
EEEEEEEEEEEE     VVV         VVV  LLL
EEE              VVV         VVV  LLL
EEE              VVV         VVV  LLL
EEE              VVV         VVV  LLL
EEE                 VVV   VVV     LLL
EEE                 VVV   VVV     LLL
EEE                 VVV   VVV     LLL
EEEEEEEEEEEEEEE        VVV        LLLLLLLLLLLLLLL
EEEEEEEEEEEEEEE        VVV        LLLLLLLLLLLLLLL
EEEEEEEEEEEEEEE        VVV        LLLLLLLLLLLLLLL
```

```
_$25

Valu
----
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
0200
```

FILEID**EVLTRANS

```
EEEEEEEEEE  VV      VV  LL          TTTTTTTTTT  RRRRRRRR      AAAAAA    NN      NN    SSSSSSSS
EEEEEEEEEE  VV      VV  LL          TTTTTTTTTT  RRRRRRRR      AAAAAA    NN      NN    SSSSSSSS
EE          VV      VV  LL              TT      RR      RR   AA    AA   NN      NN   SS
EE          VV      VV  LL              TT      RR      RR   AA    AA   NN      NN   SS
EE          VV      VV  LL              TT      RR      RR   AA    AA   NNNN    NN   SS
EE          VV      VV  LL              TT      RR      RR   AA    AA   NNNN    NN   SS
EEEEEEEE    VV      VV  LL              TT      RRRRRRRR     AA    AA   NN  NN  NN    SSSSSS
EEEEEEEE    VV      VV  LL              TT      RRRRRRRR     AA    AA   NN  NN  NN    SSSSSS
EE          VV      VV  LL              TT      RR  RR       AAAAAAAAAA   NN    NNNN          SS
EE          VV      VV  LL              TT      RR  RR       AAAAAAAAAA   NN    NNNN          SS
EE            VV  VV    LL              TT      RR    RR     AA    AA   NN      NN          SS      ....
EE            VV  VV    LL              TT      RR    RR     AA    AA   NN      NN          SS      ....
EEEEEEEEEE      VV      LLLLLLLLLL      TT      RR      RR   AA    AA   NN      NN   SSSSSSSS       ....
EEEEEEEEEE      VV      LLLLLLLLLL      TT      RR      RR   AA    AA   NN      NN   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
```

```
;  1     0001  0 %TITLE 'Event Logging Transmitter'
;  2     0002  0 MODULE EVLTRANS (IDENT = 'V04-000') =
;  3     0003  1 BEGIN
;  4     0004  1
;  5     0005  1 !
;  6     0006  1 !*****************************************************************************
;  7     0007  1 !*                                                                           *
;  8     0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;  9     0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
; 10     0010  1 !*  ALL RIGHTS RESERVED.                                                     *
; 11     0011  1 !*                                                                           *
; 12     0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
; 13     0013  1 !*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE          *
; 14     0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER     *
; 15     0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
; 16     0016  1 !*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
; 17     0017  1 !*  TRANSFERRED.                                                             *
; 18     0018  1 !*                                                                           *
; 19     0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE    *
; 20     0020  1 !*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT       *
; 21     0021  1 !*  CORPORATION.                                                             *
; 22     0022  1 !*                                                                           *
; 23     0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS     *
; 24     0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
; 25     0025  1 !*                                                                           *
; 26     0026  1 !*                                                                           *
; 27     0027  1 !*****************************************************************************
; 28     0028  1 !
; 29     0029  1
; 30     0030  1 !++
; 31     0031  1 ! FACILITY:     DECnet Event Logging (EVL)
; 32     0032  1 !
; 33     0033  1 ! ABSTRACT:
; 34     0034  1 !
; 35     0035  1 !       This module contains the routines and data which control
; 36     0036  1 !       the filtering and transmission of events to sinks on the local
; 37     0037  1 !       or remote nodes.
; 38     0038  1 !
; 39     0039  1 ! ENVIRONMENT:  VAX/VMS Operating System
; 40     0040  1 !
; 41     0041  1 ! AUTHOR:       Darrell Duffy   , CREATION DATE:  13-June-1980
; 42     0042  1 !
; 43     0043  1 ! MODIFIED BY:
; 44     0044  1 !
; 45     0045  1 !       V004    TMH0004         Tim Halvorsen   20-Jul-1983
; 46     0046  1 !               Pass our version number to remote event receiver
; 47     0047  1 !               in connect data.
; 48     0048  1 !
; 49     0049  1 !       V003    TMH0003         Tim Halvorsen   22-Nov-1982
; 50     0050  1 !               Add "area" to the list of entities which are
; 51     0051  1 !               identified before the actual parameters.
; 52     0052  1 !
; 53     0053  1 !       V002    TMH0002         Tim Halvorsen   01-Nov-1982
; 54     0054  1 !               Add "module" to the list of entities which
; 55     0055  1 !               are identified before the actual parameters.
; 56     0056  1 !
; 57     0057  1 !       V001    TMH0001         Tim Halvorsen   03-Jun-1982
```

```
;   58          0058  1 !
;   59          0059  1 !--                Convert to use new NETACP control QIO interface.
```

```
;    61          0060  1 %SBTTL  'Definitions'
;    62          0061  1
;    63          0062  1 !
;    64          0063  1 ! TABLE OF CONTENTS:
;    65          0064  1 !
;    66          0065  1
;    67          0066  1 FORWARD ROUTINE
;    68          0067  1
;    69          0068  1         EVL$TRANSMIT     : NOVALUE,      ! Initialize event transmitter
;    70          0069  1         EVL$BLDFILTERS   : NOVALUE,      ! Build sink and filter database
;    71          0070  1         EVL$BLDSOURCES   : NOVALUE,      ! Build source block on sink block
;    72          0071  1         EVL$ALLOCSNK     : NOVALUE,      ! Allocate a sink block
;    73          0072  1         EVL$DEALLOCSNK   : NOVALUE,      ! Deallocate a sink block
;    74          0073  1         EVL$OBTAINEVTS   : NOVALUE,      ! Wait on events available from NETACP
;    75          0074  1         EVL$READEVTS     : NOVALUE,      ! Read raw events from NETACP
;    76          0075  1         EVL$PROCESSEVTS  : NOVALUE,      ! Process raw event buffer
;    77          0076  1         EVL$FILTEREVT    : NOVALUE,      ! Filter a single raw event
;    78          0077  1         EVL$FILTERSRC    :,              ! Check filters with a matching source
;    79          0078  1         EVL$QUEUEVT      : NOVALUE,      ! Translate and queue an event to sink
;    80          0079  1         EVL$BLDLOSTEVENT : NOVALUE,      ! Build and queue a lost event event
;    81          0080  1         EVL$NETERROR     :,              ! Report an error in network io
;    82          0081  1         EVL$WRITEVT      : NOVALUE,      ! Write an event to sink
;    83          0082  1         EVL$WRITEDONE    : NOVALUE,      ! Complete io to sink
;    84          0083  1         EVL$OPENSNK      : NOVALUE,      ! Open a link to sink node
;    85          0084  1         EVL$OPENDONE     : NOVALUE,      ! Complete open of link
;    86          0085  1         EVL$CLOSESNK     : NOVALUE,      ! Close and possibly delete sink block
;    87          0086  1         EVL$ALLOCDBK,                    ! Allocate a data block
;    88          0087  1         EVL$DEALLOCDBK   : NOVALUE,      ! Deallocate a data block
;    89          0088  1         EVL$ASTWORK      : NOVALUE       ! AST routine to queue work request
;    90          0089  1         ;
;    91          0090  1
;    92          0091  1 !
;    93          0092  1 ! INCLUDE FILES:
;    94          0093  1 !
;    95          0094  1
;    96          0095  1 LIBRARY 'SYS$LIBRARY:STARLET';           ! VMS common definitions
;    97          0096  1 LIBRARY 'SHRLIB$:NET';                   ! Network ACP interface
;    98          0097  1 LIBRARY 'LIB$:EVLIBRARY';                ! EVL definitions
;    99          0098  1 LIBRARY 'LIB$:EVCDEF';                   ! Event record definitions
;   100          0099  1
;   101          0100  1 !
;   102          0101  1 ! EQUATED SYMBOLS:
;   103          0102  1 !
;   104          0103  1
;   105          0104  1 LITERAL
;   106          0105  1         EVL$C_EVTMBXSIZE = 64,                  ! Size of event mailbox
;   107          0106  1         EVL$C_EVTBFRSIZE = 512 - ASP$C_SIZE     ! Size of event buffer
;   108          0107  1         ;
;   109          0108  1
;   110          0109  1
;   111          0110  1 BIND
;   112          0111  1     EVL$Q_CLOSEDELAY =                          ! Delay before closing link
;   113          0112  1         UPLIT (-120*10*1000*1000, -1)           ! to event receiver
;   114          0113  1         ;
```

```
;  116     0114  1
;  117     0115  1 !
;  118     0116  1 ! OWN STORAGE:
;  119     0117  1 !
;  120     0118  1
;  121     0119  1 GLOBAL
;  122     0120  1     EVL$GQ_SNKHEAD      : VECTOR [2]              ! Head of sink node list
;  123     0121  1                 INITIAL (EVL$GQ_SNKHEAD, EVL$GQ_SNKHEAD),
;  124     0122  1     EVL$GT_LOCALNODE    : VECTOR [10, BYTE],      ! Buffer for local node info
;  125     0123  1     EVL$GB_LOCALNODE    : BYTE,                   ! Length of data in buffer
;  126     0124  1     EVL$GQ_ASPHEAD      : VECTOR [2]              ! Head of ASP queue
;  127     0125  1                 INITIAL (EVL$GQ_ASPHEAD, EVL$GQ_ASPHEAD),
;  128     0126  1     EVL$GB_TRANSDONE    : BYTE INITIAL(TRUE)      ! True for transmitter done
;  129     0127  1     ;
;  130     0128  1
;  131     0129  1 OWN
;  132     0130  1     EVL$W_NETEVTCHAN    : WORD,                   ! Network channel for events
;  133     0131  1     EVL$W_MBXEVTCHAN    : WORD                    ! Mailbox channel for events
;  134     0132  1     ;
```

```
;  136          0133  1
;  137          0134  1 !
;  138          0135  1 ! EXTERNAL REFERENCES:
;  139          0136  1 !
;  140          0137  1
;  141          0138  1 EXTERNAL LITERAL
;  142          0139  1
;  143          0140  1         EVL$_NETASN,            ! Error assigning to NET
;  144          0141  1         EVL$_OBTEVT,            ! Error obtaining raw events
;  145          0142  1         EVL$_RAWFMT,            ! Invalid format for raw event
;  146          0143  1         EVL$_WRTEVT,            ! Error writing events to sink node
;  147          0144  1         EVL$_OPNSNK,            ! Error opening sink node link
;  148          0145  1         EVL$_NETDAS,            ! Error deassigning NET channel
;  149          0146  1         EVL$_INSFVM,            ! Unable to obtain virtual memory
;  150          0147  1         EVL$_EVTSIZ,            ! Invalid size of event
;  151          0148  1         EVL$_WKQERR,            ! Error returned from work queue routines
;  152          0149  1         EVL$_LOGDBUT,           ! Log of data base updates by transmitter
;  153          0150  1         EVL$_LOGOPNT,           ! Log of open links by transmitter
;  154          0151  1         EVL$_NETSHUT            ! Network is shutting down
;  155          0152  1         ;
;  156          0153  1
;  157          0154  1 EXTERNAL
;  158          0155  1         EVL$GL_LOGMASK : BBLOCK ! Log control mask
;  159          0156  1         ;
;  160          0157  1
;  161          0158  1 EXTERNAL ROUTINE
;  162          0159  1
;  163          0160  1         EVL$JULIAN              : NOVALUE,      ! Convert time to julian
;  164          0161  1         WKQ$ADD_WORK_ITEM       : ,             ! Add element to work queue
;  165          0162  1         WKQ$ADD_TIMED_WORK      : ,             ! Add scheduled work to queue
;  166          0163  1         WKQ$CANCEL_TIMED_WORK   : ,             ! Cancel timed work item
;  167          0164  1         EVL$QUEUE_EVENT         . NOVALUE,      ! Queue event to local receiver
;  168          0165  1         EVL$NETSHOW             : ,             ! Show data from NETACP
;  169          0166  1         EVL$PRINTLOG            : NOVALUE,      ! Print message to log
;  170          0167  1         EVL$INITLOCALNODE       : NOVALUE       ! Initialize local node name
;  171          0168  1         ;
;  172          0169  1
;  173          0170  1 !
;  174          0171  1 !       Library routines
;  175          0172  1 !
;  176          0173  1
;  177          0174  1 EXTERNAL ROUTINE
;  178          0175  1
;  179          0176  1         LIB$GET_VM : ADDRESSING_MODE (GENERAL),
;  180          0177  1         LIB$FREE_VM : ADDRESSING_MODE (GENERAL),
;  181          0178  1         LIB$ASN_WTH_MBX : ADDRESSING_MODE (GENERAL)
;  182          0179  1         ;
```

```
;  184          0180  1 %SBTTL  'EVL$TRANSMIT  Main Initialization'
;  185          0181  1 GLOBAL ROUTINE EVL$TRANSMIT :NOVALUE =
;  186          0182  1
;  187          0183  1 !++
;  188          0184  1 ! FUNCTIONAL DESCRIPTION:
;  189          0185  1 !
;  190          0186  1 !       Main initialization routine for event logg ng transmitter.
;  191          0187  1 !       Calls the routines to initialize the data base here for the
;  192          0188  1 !       transmitter and start the event logging.  After this routine
;  193          0189  1 !       returns to its caller, work queue elements are on the work queue
;  194          0190  1 !       to continue the event logging.
;  195          0191  1 !
;  196          0192  1 ! FORMAL PARAMETERS:
;  197          0193  1 !
;  198          0194  1 !       NONE
;  199          0195  1 !
;  200          0196  1 ! IMPLICIT INPUTS:
;  201          0197  1 !
;  202          0198  1 !       NONE
;  203          0199  1 !
;  204          0200  1 ! IMPLICIT OUTPUTS:
;  205          0201  1 !
;  206          0202  1 !       NONE
;  207          0203  1 !
;  208          0204  1 ! ROUTINE VALUE:
;  209          0205  1 ! COMPLETION CODES:
;  210          0206  1 !
;  211          0207  1 !       NONE
;  212          0208  1 !
;  213          0209  1 ! SIDE EFFECTS:
;  214          0210  1 !
;  215          0211  1 !       NONE
;  216          0212  1 !
;  217          0213  1 !--
;  218          0214  1
;  219          0215  2     BEGIN
;  220          0216  2
;  221          0217  2     LOCAL
;  222          0218  2         STATUS,                             ! Local status
;  223          0219  2         IOSB     : VECTOR [2],              ! Iosb for simple qio
;  224          0220  2         MBXMODES : VECTOR [2]               ! Enable mailbox modes
;  225          0221  2         ;
;  226          0222  2
;  227          0223  2     EVL$INITLOCALNODE ();       ! Initialize the local node name
;  228          0224  2
;  229          0225  2     IF NOT
;  230          0226  3         ( STATUS = LIB$ASN_WTH_MBX      ! Get some channels for use
;  231          0227  3             (
;  232          0228  3             %ASCID ' NET:',             ! To network
;  233          0229  3             0, %REF (EVL$C_EVTMBXSIZE), ! Size of mailbox
;  234          0230  3             EVL$W_NETEVTCHAN,           ! Network channel
;  235          0231  3             EVL$W_MBXEVTCHAN            ! Mailbox channel
;  236          0232  3             )
;  237          0233  3         )
;  238          0234  2     THEN
;  239          0235  2         SIGNAL_STOP (EVL$_NETASN, 0, .STATUS)   ! Stop right now
;  240          0236  2     ;
```

```
;  241          0237  2
;  242          0238  2      MBXMODES [0] = 0;                        ! Enable all modes
;  243          0239  2      MBXMODES [1] = -1;
;  244          0240  2
;  245        P 0241  2      STATUS = $QIOW                           ! Set mailbox modes
;  246        P 0242  2          (
;  247        P 0243  2          EFN = EVL$C_SYNCH_EFN,
;  248        P 0244  2          CHAN = .EVL$W_NETEVTCHAN,
;  249        P 0245  2          FUNC = IO$_SETMODE,
;  250        P 0246  2          IOSB = IOSB,
;  251        P 0247  2          P1 = MBXMODES
;  252          0248  2          );
;  253          0249  2
;  254          0250  2      EVL$NETERROR (EVL$_OBTEVT, .STATUS, IOSB);  ! Analyse errors
;  255          0251  2
;  256          0252  2      EVL$BLDFILTERS ();          ! Build our filter database
;  257          0253  2      EVL$OBTAINEVTS (0)          ! Obtain the events and start processing
;  258          0254  2
;  259          0255  1      END;


                                                .TITLE  EVLTRANS Event Logging Transmitter
                                                .IDENT  \V04-000\

                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                        FFFFFFFF  B8797400  00000 P.AAA:  .LONG   -1200000000, -1
                00  00  00  3A  54  45  4E  5F  00008 P.AAC:  .ASCII  \_NET:\<0><0><0>                    ;
                                  010E0005  00010 P.AAB:  .LONG   17694725                              ;
                                  00000000' 00014         .ADDRESS P.AAC                                ;

                                                .PSECT  $OWN$,NOEXE,2

                                        00000 EVL$W_NETEVTCHAN:
                                                .BLKB   2
                                        00002 EVL$W_MBXEVTCHAN:
                                                .BLKB   2

                                                .PSECT  $GLOBAL$,NOEXE,2

        00000000' 00000000' 00000 EVL$GQ_SNKHEAD::
                                                .ADDRESS EVL$GQ_SNKHEAD, EVL$GQ_SNKHEAD                 ;
                                        00008 EVL$GT_LOCALNODE::
                                                .BLKB   10
                                        00012 EVL$GB_LOCALNODE::
                                                .BLKB   1
                                        00013           .BLKB   1
        00000000' 00000000' 00014 EVL$GQ_ASPHEAD::
                                                .ADDRESS EVL$GQ_ASPHEAD, EVL$GQ_ASPHEAD                 ;
                             01  0001C EVL$GB_TRANSDONE::
                                                .BYTE   1                                               ;

                                    EVL$Q_CLOSEDELAY=   P.AAA
                                                .EXTRN  EVL$_NETASN, EVL$_OBTIL
                                                .EXTRN  EVL$_RAWFMT, EVL$_WRTEVT
                                                .EXTRN  EVL$_OPNSNK, EVL$_NETDAS
                                                .EXTRN  EVL$_INSFVM, EVL$_EVTSIZ
```

```
                                        .EXTRN  EVL$_WKQERR, EVL$_LOGDBUT
                                        .EXTRN  EVL$_LOGOPNT, EVL$_NETSHUT
                                        .EXTRN  EVL$GL_LOGMASK, EVC$JULIAN
                                        .EXTRN  WKQ$ADD_WORK_ITEM
                                        .EXTRN  WKQ$ADD_TIMED_WORK
                                        .EXTRN  WKQ$CANCEL_TIMED_WORK
                                        .EXTRN  EVL$QUEUE_EVENT
                                        .EXTRN  EVL$NETSHOW, EVL$PRINTLOG
                                        .EXTRN  EVL$INITLOCALNODE
                                        .EXTRN  LIB$GET_VM, LIB$FREE_VM
                                        .EXTRN  LIB$ASN_WTH_MBX
                                        .EXTRN  SYS$QIOW

                                        .PSECT  $CODE$,NOWRT,2

                                0004 00000      .ENTRY  EVL$TRANSMIT, Save R2                   ; 0181
                             5E         14  C2 00002      SUBL2   #20, SP                         ;
                   0000G     CF         00  FB 00005      CALLS   #0, EVL$INITLOCALNODE           ; 0223
                                 0000'  CF  9F 0000A      PUSHAB  EVL$W_MBXEVTCHAN                ; 0227
                                 0000'  CF  9F 0000E      PUSHAB  EVL$W_NETEVTCHAN                ;
                        08       AE    40  8F 9A 00012    MOVZBL  #64, 8(SP)                      ; 0229
                                       08  AE 9F 00017    PUSHAB  8(SP)                           ;
                                           7E D4 0001A    CLRL    -(SP)                           ; 0227
                                 0000'  CF  9F 0001C      PUSHAB  P.AAB                           ;
                00000000G        00         05  FB 00020  CALLS   #5, LIB$ASN_WTH_MBX            ;
                                 52         50  D0 00027  MOVL    R0, STATUS                    ;
                                 11         52  E8 0002A  BLBS    STATUS, 1$                    ;
                                            52  DD 0002D  PUSHL   STATUS                        ; 0235
                                            7E  D4 0002F  CLRL    -(SP)                         ;
                                   00000000G    8F  DD 00031  PUSHL   #EVL$_NETASN              ;
                00000000G        00         03  FB 00037  CALLS   #3, LIB$STOP                 ;
                                       04  AE  D4 0003E  1$:  CLRL    MBXMODES                  ; 0238
                             08    AE         01  CE 00041  MNEGL   #1, MBXMODES+4             ; 0239
                                              7E  7C 00045  CLRQ    -(SP)                       ; 0248
                                              7E  7C 00047  CLRQ    -(SP)                       ;
                                              7E  D4 00049  CLRL    -(SP)                       ;
                                       18  AE  9F 0004B    PUSHAB  MBXMODES                    ;
                                              7E  7C 0004E  CLRQ    -(SP)                       ;
                                       2C  AE  9F 00050    PUSHAB  IOSB                        ;
                                           23  DD 00053    PUSHL   #35                         ;
                                 7E    0000' CF  3C 00055  MOVZWL  EVL$W_NETEVTCHAN, -(SP)     ;
                                           01  DD 0005A    PUSHL   #1                          ;
                00000000G        00         0C  FB 0005C  CALLS   #12, SYS$QIOW               ;
                                 52         50  D0 00063  MOVL    R0, STATUS                  ;
                                       0C  AE  9F 00066    PUSHAB  IOSB                        ; 0250
                                           52  DD 00069    PUSHL   STATUS                      ;
                                   00000000G    8F  DD 0006B  PUSHL   #EVL$_OBTEVT              ;
                   0000V     CF         03  FB 00071      CALLS   #3, EVL$NETERROR              ;
                   0000V     CF         00  FB 00076      CALLS   #0, EVL$BLDFILTERS            ; 0252
                                           7E  D4 0007B    CLRL    -(SP)                       ; 0253
                   0000V     CF         01  FB 0007D      CALLS   #1, EVL$OBTAINEVTS            ;
                                           04  00082      RET                                 ; 0255

; Routine Size:  131 bytes,    Routine Base:  $CODE$ + 0000
```

```
;   261          0256  1 %SBTTL  'EVL$BLDFILTERS  Build Our Local Database'
;   262          0257  1 ROUTINE EVL$BLDFILTERS :NOVALUE =
;   263          0258  1
;   264          0259  1 !++
;   265          0260  1 ! FUNCTIONAL DESCRIPTION:
;   266          0261  1 !
;   267          0262  1 !       Update the local database for the event transmitter.  If a sink node
;   268          0263  1 !       is not present it is added to the queue.  All source filters on all
;   269          0264  1 !       sinks are replaced with the latest data.  Sinks that are not present
;   270          0265  1 !       are either deallocated outright, or if there are events in the queue,
;   271          0266  1 !       marked for later close and deallocate.
;   272          0267  1 !
;   273          0268  1 !       If at the end of this routine, there are no sink blocks on the queue,
;   274          0269  1 !       we set a flag saying we have no work to do.  Otherwise we set the
;   275          0270  1 !       flag saying that we have work to do.
;   276          0271  1 !
;   277          0272  1 ! FORMAL PARAMETERS:
;   278          0273  1 !
;   279          0274  1 !       NONE
;   280          0275  1 !
;   281          0276  1 ! IMPLICIT INPUTS:
;   282          0277  1 !
;   283          0278  1 !       NETACP logging database
;   284          0279  1 !
;   285          0280  1 ! IMPLICIT OUTPUTS:
;   286          0281  1 !
;   287          0282  1 !       Sink node queue updated
;   288          0283  1 !
;   289          0284  1 ! ROUTINE VALUE:
;   290          0285  1 ! COMPLETION CODES:
;   291          0286  1 !
;   292          0287  1 !       NONE
;   293          0288  1 !
;   294          0289  1 ! SIDE EFFECTS:
;   295          0290  1 !
;   296          0291  1 !       NONE
;   297          0292  1 !
;   298          0293  1 !--
;   299          0294  1
;   300          0295  2     BEGIN
;   301          0296  2
;   302          0297  2     LITERAL
;   303          0298  2         SNKBFRSIZE = 512                ! Size of sink buffer
;   304          0299  2         ;
;   305          0300  2
;   306          0301  2     LOCAL
;   307          0302  2         SNKQUE,                         ! Pointer to sink queue head
;   308          0303  2         POSITION:VECTOR [NFB$C_CTX_SIZE,BYTE],  ! Current EFI position
;   309          0304  2         SNKBFR : BBLOCK [SNKBFRSIZE],   ! Buffer for sink data
;   310          0305  2         SNKDSC : VECTOR [2],            ! Descriptor of buffer
;   311          0306  2         SNKPTR : REF BBLOCK;            ! Pointer to sink block
;   312          0307  2
;   313          0308  2     SNKQUE = EVL$GQ_SNKHEAD;            ! Head of sink queue
;   314          0309  2     SNKPTR = ..SNKQUE;                  ! Setup pointer
;   315          0310  2
;   316          0311  2     WHILE .SNKPTR NEQ .SNKQUE           ! Scan current queue
;   317          0312  2     DO
```

```
;  318     0313  3          BEGIN
;  319     0314  3          SNKPTR [SNK$V_STS_DEL] = TRUE;  ! Set deleted bit in all
;  320     0315  3          SNKPTR = .SNKPTR [SNK$L_FL]     ! Link to next
;  321     0316  3          END
;  322     0317  2      ;
;  323     0318  2
;  324     0319  2      POSITION <0,16> = 0;                ! Start at first EFI record
;  325     0320  2
;  326     0321  2      SNKDSC [0] = SNKBFRSIZE;            ! Descriptor of return buffer
;  327     0322  2      SNKDSC [1] = SNKBFR;
;  328     0323  2
;  329     0324  2      WHILE                               ! As long as there are more nodes
;  330     0325  2          EVL$NETSHOW(                    ! Obtain a sink node
;  331     0326  2              NFB$C_DB_EFI,               ! Event information
;  332     0327  2              NFB$C_WILDCARD, 0,          ! No search key
;  333     0328  2              POSITION,                   ! Get next record, update position
;  334     0329  2              2, UPLIT (NFB$C_EFI_SIN, NFB$C_EFI_EVE),
;  335     0330  2              SNKDSC)                     ! Buffer descriptor
;  336     0331  2      DO
;  337     0332  3          BEGIN
;  338     0333  3          SNKPTR = ..SNKQUE;              ! Scan the queue for the new sink
;  339     0334  3          IF
;  340     0335  4              BEGIN
;  341     0336  4              WHILE .SNKPTR NEQ .SNKQUE   ! Scan to the end
;  342     0337  4              DO
;  343     0338  5                  BEGIN
;  344     0339  5                  IF .SNKBFR [0,0,32,0] EQL .SNKPTR [SNK$L_SNKADR]        ! Do the addresses
;  345     0340  5                  THEN                                                    ! match?
;  346     0341  6                      BEGIN
;  347     0342  6                      SNKPTR [SNK$V_STS_DEL] = FALSE;     ! Its here, so keep it
;  348     0343  6                      SNKPTR [SNK$V_STS_CLS] = FALSE;
;  349     0344  6                      EVL$BLDSOURCES (.SNKPTR, SNKBFR+4); ! Build rest of data
;  350     0345  6                      EXITLOOP FALSE                      ! Do not create one
;  351     0346  6                      END
;  352     0347  5                  ;
;  353     0348  5                  SNKPTR = .SNKPTR [SNK$L_FL]             ! Link to next
;  354     0349  5                  END
;  355     0350  4              END
;  356     0351  3          THEN
;  357     0352  4              BEGIN
;  358     0353  4              EVL$ALLOCSNK (.SNKQUE, SNKPTR);             ! Allocate a new snk
;  359     0354  4              SNKPTR [SNK$L_SNKADR] = .SNKBFR [0,0,32,0]; ! Save the sink addr
;  360     0355  4              SNKPTR [SNK$L_SNKLEN] = SNK$S_SNKNCB;       ! Build ncb dsc
;  361     0356  4              SNKPTR [SNK$A_SNKNCB] = SNKPTR [SNK$T_SNKNCB];
;  362   P 0357  4              $FAOL                                       ! Build the ncb
;  363   P 0358  4                  (                                       ! in the sink block
;  364   P 0359  4                  CTRSTR = %ASCID '!UL::''26=/',          ! Use the address form
;  365   P 0360  4                  OUTLEN = SNKPTR [SNK$L_SNKLEN],         ! of the connect
;  366   P 0361  4                  OUTBUF = SNKPTR [SNK$L_SNKLEN],
;  367   P 0362  4                  PRMLST = SNKBFR
;  368     0363  4                  );
;  369     0364  4              CH$MOVE(20, UPLIT BYTE(                     ! Append:
;  370     0365  4                          0,0,                            ! Link number
;  371     0366  4                          3, 4, 0,0,                      ! Our version number
;  372     0367  4                          REP 13 OF (0),                  ! Pad out unused data
;  373     0368  4                          %"")                            ! and NCB terminator
;  374     0369  4                  .SNKPTR [SNK$A_SNKNCB] + .SNKPTR [SNK$L_SNKLEN]);
```

```
;  375     0370  4              SNKPTR [SNK$L_SNKLEN] = .SNKPTR [SNK$L_SNKLEN]+20; ! Adjust length
;  376     0371  4              EVL$BLDSOURCES (.SNKPTR, SNKBFR+4)          ! and fill it in
;  377     0372  4              END
;  378     0373  3          END
;  379     0374  2      ;
;  380     0375  2
;  381     0376  2  !
;  382     0377  2  !       Scan the snk list and remove deleted nodes
;  383     0378  2  !
;  384     0379  2
;  385     0380  2      SNKPTR = ..SNKQUE;
;  386     0381  2      WHILE .SNKPTR NEQ .SNKQUE                   ! Scan the list
;  387     0382  2      DO
;  388     0383  3          BEGIN
;  389     0384  3          IF .SNKPTR [SNK$V_STS_DEL]              ! If deleted,
;  390     0385  3              AND NOT .SNKPTR [SNK$V_STS_CLS]     ! and not already being closed,
;  391     0386  3          THEN
;  392     0387  4              BEGIN
;  393     0388  4              SNKPTR [SNK$V_STS_CLS] = TRUE;      ! Set close as true
;  394     0389  4              WKQ$ADD_WORK_ITEM(EVL$WRITEVT,.SNKPTR); ! Make sure I/O going
;  395     0390  4                                                  ! (will delete if I/O done)
;  396     0391  4              IF .EVL$GL_LOGMASK [ELG$V_DBUPDAT]  ! Log data base updates?
;  397     0392  4              THEN
;  398     0393  4                  SIGNAL (EVL$_LOGDBUT, 1, 0)     ! Log that it occurred
;  399     0394  3              END;
;  400     0395  3          SNKPTR = .SNKPTR [SNK$L_FL];            ! Next snk in list
;  401     0396  2          END;
;  402     0397  2
;  403     0398  1      END;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                                 06020041  06010010  00018 P.AAD:  .LONG   100728848, 100794433
00  00  2F  3D  36  32  22  3A  3A  4C  55  21  00020 P.AAF:  .ASCII  \!UL::''26=/\<0><0>
                             010E000A  0002C P.AAE:  .LONG   17694730
                             00000000' 00030         .ADDRESS P.AAF
                         00  00  04  03  00  00  00034 P.AAG:  .BYTE   0, 0, 3, 4, 0, 0
                                             00#  0003A         .BYTE   0[13]
                                             22   00047         .ASCII  \''\

                                                        .EXTRN  SYS$FAOL

                                                        .PSECT  $CODE$,NOWRT,2

                                         01FC 00000 EVL$BLDFILTERS:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8           ; 0257
                          5E      FDB8   CE   9E 00002          MOVAB   -584(SP), SP
                          58      0000'  CF   9E 00007          MOVAB   EVL$GQ_SNKHEAD, SNKQUE      ; 0308
                                         68   DD 0000C          PUSHL   (SNKQUE)                    ; 0309
                          50             6E   D0 0000E 1$:      MOVL    SNKPTR, R0                  ; 0311
                          58             50   D1 00011          CMPL    R0, SNKQUE
                                         09   13 00014          BEQL    2$
                      2E  A0             08   88 00016          BISB2   #8, 46(R0)                  ; 0314
                          6E             60   D0 0001A          MOVL    (R0), SNKPTR                ; 0315
                                         EF   11 0001D          BRB     1$
```

```
                                 C0   AD   B4 0001F 2$:     CLRW    POSITION                                ; 0319
                              04 AE   0200   8F   3C 00022         MOVZWL  #512, SNKDSC                    ; 0321
                              08 AE     0C   AE   9E 00028         MOVAB   SNKBFR, SNKDSC+4                ; 0322
                                 04     AE   9F 0002D 3$:     PUSHAB  SNKDSC                              ; 0325
                                 0000'  CF   9F 00030         PUSHAB  P.AAD                               ; 0329
                                        02   DD 00034         PUSHL   #2                                  ; 0325
                                 C0     AD   9F 00036         PUSHAB  POSITION                            ;
                              7E                01   7D 00039         MOVQ    #1, -(SP)                   ;
                                        06   DD 0003C         PUSHL   #6                                  ;
                           0000G  CF            07   FB 0003E         CALLS   #7, EVL$NETSHOW             ;
                              6D                50   E9 00043         BLBC    R0, 8$                      ;
                              6E                68   D0 00046         MOVL    (SNKQUE), SNKPTR            ; 0333
                              52                6E   D0 00049 4$:     MOVL    SNKPTR, R2                  ; 0336
                              58                52   D1 0004C         CMPL    R2, SNKQUE                  ;
                                                17   13 0004F         BEQL    6$                          ;
                              18   A2     0C   AE   D1 00051         CMPL    SNKBFR, 24(R2)              ; 0339
                                                0B   12 00056         BNEQ    5$                          ;
                              2E   A2           18   8A 00058         BICB2   #24, 46(R2)                 ; 0343
                                 10     AE   9F 0005C         PUSHAB  SNKBFR+4                            ; 0344
                                        52   DD 0005F         PUSHL   R2                                  ;
                                        48   11 00061         BRB     7$                                  ;
                              6E                62   D0 00063 5$:     MOVL    (R2), SNKPTR                ; 0348
                                        E1   11 00066         BRB     4$                                  ;
                                 4100   8F   BB 00068 6$:     PUSHR   #^M<R8,SP>                          ; 0353
                           0000V  CF            02   FB 0006C         CALLS   #2, EVL$ALLOCSNK            ;
                              56                6E   D0 00071         MOVL    SNKPTR, R6                  ; 0354
                              18   A6     0C   AE   D0 00074         MOVL    SNKBFR, 24(R6)              ;
                              57   A6     30   A6   9E 00079         MOVAB   48(R6), R7                  ; 0355
                              67           40   8F   9A 0007D         MOVZBL  #64, (R7)                   ;
                              34   A6     38   A6   9E 00081         MOVAB   56(R6), 52(R6)              ; 0356
                                 0C     AE   9F 00086         PUSHAB  SNKBFR                              ; 0363
                                        57   DD 00089         PUSHL   R7                                  ;
                                        57   DD 0008B         PUSHL   R7                                  ;
                                 0000'  CF   9F 0008D         PUSHAB  P.AAE                               ;
                        00000000G  00            04   FB 00091         CALLS   #4, SYS$FAOL                ;
                           50   34   A6           67   C1 00098         ADDL3   (R7), 52(R6), R0            ; 0369
                           60   0000' CF           14   28 0009D         MOVC3   #20, P.AAG, (R0)            ;
                              67                14   C0 000A3         ADDL2   #20, (R7)                   ; 0370
                                    10   AE   9F 000A6         PUSHAB  SNKBFR+4                            ; 0371
                                        56   DD 000A9         PUSHL   R6                                  ;
                           0000V  CF            02   FB 000AB 7$:     CALLS   #2, EVL$BLDSOURCES          ;
                                       FF7A   31   B0 000B0         BRW     3$                              ; 0334
                              6E                68   D0 000B3 8$:     MOVL    (SNKQUE), SNKPTR            ; 0380
                              52                6E   D0 000B6 9$:     MOVL    SNKPTR, R2                  ; 0381
                              58                52   D1 000B9         CMPL    R2, SNKQUE                  ;
                                                33   13 000BC         BEQL    11$                         ;
                           29   2E   A2           03   E1 000BE         BBC     #3, 46(R2), 10$             ; 0384
                           24   2E   A2           04   E0 000C3         BBS     #4, 46(R2), 10$             ; 0385
                              2E   A2           10   88 000C8         BISB2   #16, 46(R2)                 ; 0388
                                        52   DD 000CC         PUSHL   R2                                  ; 0389
                                 0000V  CF   9F 000CE         PUSHAB  EVL$WRITEVT                         ;
                           0000G  CF            02   FB 000D2         CALLS   #2, WKQ$ADD_WORK_ITEM       ;
                              10           0000G  CF   E9 000D7         BLBC    EVL$GL_LOGMASK, 10$         ; 0391
                              7E                01   7D 000DC         MOVQ    #1, -(SP)                   ; 0393
                                    00000000G  8F   DD 000DF         PUSHL   #EVL$_LOGDBUT               ;
                        00000000G  00            03   FB 000E5         CALLS   #3, LIB$SIGNAL              ;
                              6E                62   D0 000EC 10$:     MOVL    (R2), SNKPTR                ; 0395
```

```
                                C5  11 000EF                 BRB     9$                                      ; 0381
                                    04 000F1 11$:           RET                                             ; 0398

; Routine Size:  242 bytes,    Routine Base:  $CODE$ + 0083
```

```
;  405     0399  1 %SBTTL  'EVL$BLDSOURCES  Build List of Sources in SNK Block'
;  406     0400  1 ROUTINE EVL$BLDSOURCES (SNKBLK, SRCDATA) :NOVALUE =
;  407     0401  1
;  408     0402  1 !++
;  409     0403  1 ! FUNCTIONAL DESCRIPTION:
;  410     0404  1 !
;  411     0405  1 !       Build source filter list on sink block.  All present source blocks
;  412     0406  1 !       are removed first.
;  413     0407  1 !
;  414     0408  1 ! FORMAL PARAMETERS:
;  415     0409  1 !
;  416     0410  1 !       SNKBLK          Address of sink block
;  417     0411  1 !       SRCDATA         Address of source data list
;  418     0412  1 !
;  419     0413  1 ! IMPLICIT INPUTS:
;  420     0414  1 !
;  421     0415  1 !       NONE
;  422     0416  1 !
;  423     0417  1 ! IMPLICIT OUTPUTS:
;  424     0418  1 !
;  425     0419  1 !       NONE
;  426     0420  1 !
;  427     0421  1 ! ROUTINE VALUE:
;  428     0422  1 ! COMPLETION CODES:
;  429     0423  1 !
;  430     0424  1 !       NONE
;  431     0425  1 !
;  432     0426  1 ! SIDE EFFECTS:
;  433     0427  1 !
;  434     0428  1 !       NONE
;  435     0429  1 !
;  436     0430  1 !--
;  437     0431  1
;  438     0432  2     BEGIN
;  439     0433  2
;  440     0434  2     LOCAL
;  441     0435  2         SNK : REF BBLOCK,               ! Pointer to sink block
;  442     0436  2         SRC : REF BBLOCK,               ! Pointer to source block
;  443     0437  2         SRCPTR,                         ! Pointer to source data
;  444     0438  2         SRCSIZE,                        ! Size of a source block
;  445     0439  2         SRCEND                          ! End of source data
;  446     0440  2         ;
;  447     0441  2
;  448     0442  2     SNK = .SNKBLK;                      ! Deallocate all current sources
;  449     0443  2     UNTIL SNK [SNK$L_SRCFL] EQL .SNK [SNK$L_SRCFL]
;  450     0444  2     DO
;  451     0445  2         EVL$DEALLOCDBK (.SNK [SNK$L_SRCFL] )
;  452     0446  2     ;
;  453     0447  2
;  454     0448  2     SRCEND = .SRCDATA + .(.SRCDATA) <0, 16>; ! Word counted string
;  455     0449  2     SRCPTR = .SRCDATA + 2;              ! Skip the count
;  456     0450  2
;  457     0451  2     WHILE .SRCPTR LSSU .SRCEND          ! For as long as there is data
;  458     0452  2     DO
;  459     0453  3         BEGIN
;  460     0454  3         SRCSIZE = .(.SRCPTR) <0, 16>;   ! Size of source block
;  461     0455  3         EVL$ALLOCDBK                    ! Build a block for the source data
```

```
;  462          0456  3                 (
;  463          0457  3                 .SRCSIZE + 8,               ! Size of source block and links
;  464          0458  3                 .SNK [SNK$L_SRCBL],         ! Link it in here
;  465          0459  3                 SRC                         ! Return its address here
;  466          0460  3                 );
;  467          0461  3             CH$MOVE                         ! Copy the data into the source block
;  468          0462  3                 (
;  469          0463  3                 .SRCSIZE,
;  470          0464  3                 .SRCPTR,
;  471          0465  3                 SRC [SRC$W_SIZE]
;  472          0466  3                 );
;  473          0467  3             SRCPTR = .SRCPTR + .SRCSIZE     ! Next source block
;  474          0468  3             END
;  475          0469  3
;  476          0470  1         END;



                                          03FC 00000 EVL$BLDSOURCES:
                                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9                          ; 0400
                         5E              04  C2 00002           SUBL2   #4, SP                                                 ;
                         56        04    AC  D0 00005           MOVL    SNKBLK, SNK                                            ; 0442
                         50        1C    A6  9E 00009 1$:       MOVAB   28(SNK), R0                                            ; 0443
                 1C      A6              50  D1 0000D           CMPL    R0, 28(SNK)                                            ;
                                         0A  13 00011           BEQL    2$                                                     ;
                                   1C    A6  DD 00013           PUSHL   28(SNK)                                                ; 0445
               0000V     CF              01  FB 00016           CALLS   #1, EVL$DEALLOCDBK                                     ;
                                         EC  11 0001B           BRB     1$                                                     ;
                         59        08    BC  3C 0001D 2$:       MOVZWL  @SRCDATA, SRCEND                                       ; 0448
                         59        08    AC  C0 00021           ADDL2   SRCDATA, SRCEND                                        ;
          58       08    AC              02  C1 00025           ADDL3   #2, SRCDATA, SRCPTR                                    ; 0449
                         59              58  D1 0002A 3$:       CMPL    SRCPTR, SRCEND                                         ; 0451
                                         1D  1E 0002D           BGEQU   4$                                                     ;
                         57              68  3C 0002F           MOVZWL  (SRCPTR), SRCSIZE                                      ; 0454
                                         5E  DD 00032           PUSHL   SP                                                     ; 0456
                                   20    A6  DD 00034           PUSHL   32(SNK)                                                ; 0458
                                   08    A7  9F 00037           PUSHAB  8(SRCSIZE)                                             ; 0457
               0000V     CF              03  FB 0003A           CALLS   #3, EVL$ALLOCDBK                                       ;
                         50              6E  D0 0003F           MOVL    SRC, R0                                                ; 0465
     08     A0           68              57  28 00042           MOVC3   SRCSIZE, (SRCPTR), 8(R0)                               ;
                         58              57  C0 00047           ADDL2   SRCSIZE, SRCPTR                                        ; 0467
                                         DE  11 0004A           BRB     3$                                                     ;
                                             04 0004C 4$:       RET                                                            ; 0470

; Routine Size:  77 bytes,    Routine Base:  $CODE$ + 0175
```

```
;  478          0471  1 %SBTTL  'EVL$ALLUCSNK  Allocate and Initialize a SNK Block'
;  479          0472  1 GLOBAL ROUTINE EVL$ALLOCSNK (PRED, SNKRTN) :NOVALUE =
;  480          0473  1
;  481          0474  1 !++
;  482          0475  1 ! FUNCTIONAL DESCRIPTION:
;  483          0476  1 !
;  484          0477  1 !       Allocate and initialize a sink block.  The block is linked into the
;  485          0478  1 !       correct place in the sink queue.
;  486          0479  1 !
;  487          0480  1 ! FORMAL PARAMETERS:
;  488          0481  1 !
;  489          0482  1 !       PRED            Address of predecessor in list
;  490          0483  1 !       SNKRTN          Address to return address of snk block
;  491          0484  1 !
;  492          0485  1 ! IMPLICIT INPUTS:
;  493          0486  1 !
;  494          0487  1 !       NONE
;  495          0488  1 !
;  496          0489  1 ! IMPLICIT OUTPUTS:
;  497          0490  1 !
;  498          0491  1 !       NONE
;  499          0492  1 !
;  500          0493  1 ! ROUTINE VALUE:
;  501          0494  1 ! COMPLETION CODES:
;  502          0495  1 !
;  503          0496  1 !       NONE
;  504          0497  1 !
;  505          0498  1 ! SIDE EFFECTS:
;  506          0499  1 !
;  507          0500  1 !       NONE
;  508          0501  1 !
;  509          0502  1 !--
;  510          0503  1
;  511          0504  2     BEGIN
;  512          0505  2
;  513          0506  2     LOCAL
;  514          0507  2         SNK : REF BBLOCK
;  515          0508  2         ;
;  516          0509  2
;  517          0510  2     EVL$ALLOCDBK (SNK$C_SIZE, .PRED, SNK);      ! Allocate the block
;  518          0511  2     SNK [SNK$L_SRCFL] = SNK [SNK$L_SRCFL];      ! Fill in the queue headers
;  519          0512  2     SNK [SNK$L_SRCBL] = SNK [SNK$L_SRCFL];
;  520          0513  2     SNK [SNK$L_EVTFL] = SNK [SNK$L_EVTFL];
;  521          0514  2     SNK [SNK$L_EVTBL] = SNK [SNK$L_EVTFL];
;  522          0515  2     .SNKRTN = .SNK;                             ! Return the sink address
;  523          0516  2
;  524          0517  2     EVL$GB_TRANSDONE = FALSE;                   ! Mark transmitter active
;  525          0518  2
;  526          0519  1     END;


                                    0000 00000          .ENTRY  EVL$ALLOCSNK, Save nothing      ; 0472
                              5E    04  C2 00002          SUBL2   #4, SP                          ;
                                    5E  DD 00005          PUSHL   SP                              ; 0510
```

```
                                04    AC  DD 00007        PUSHL   PRED                          ;
                  7E            78    8F  9A 0000A        MOVZBL  #120, -(SP)                   ;
          0000V   CF                  03  FB 0000E        CALLS   #3, EVL$ALLOCDBK              ;
                  50                  6E  D0 00013        MOVL    SNK, R0                       ; 0511
            1C    A0            1C    A0  9E 00016        MOVAB   28(R0), 28(R0)                ;
            20    A0            1C    A0  9E 0001B        MOVAB   28(R0), 32(R0)                ; 0512
            24    A0            24    A0  9E 00020        MOVAB   36(R0), 36(R0)                ; 0513
            28    A0            24    A0  9E 00025        MOVAB   36(R0), 40(R0)                ; 0514
            08    BC                  50  D0 0002A        MOVL    R0, @SNKRTN                   ; 0515
                          0000'   CF  94 0002E        CLRB    EVL$GB_TRANSDONE              ; 0517
                                      04 00032        RET                                   ; 0519

; Routine Size:  51 bytes,    Routine Base:  $CODE$ + 01C2
```

```
;   528          0520  1 %SBTTL  'EVL$DEALLOCSNK  Deallocate Sink Block'
;   529          0521  1 ROUTINE EVL$DEALLOCSNK (SNKBLK) :NOVALUE =
;   530          0522  1
;   531          0523  1 !++
;   532          0524  1 ! FUNCTIONAL DESCRIPTION:
;   533          0525  1 !
;   534          0526  1 !       Deallocate a sink block.  Remove and deallocate all the entries
;   535          0527  1 !       on the src and evt queues as well.
;   536          0528  1 !
;   537          0529  1 ! FORMAL PARAMETERS:
;   538          0530  1 !
;   539          0531  1 !       SNKBLK          Address of the sink block
;   540          0532  1 !
;   541          0533  1 ! IMPLICIT INPUTS:
;   542          0534  1 !
;   543          0535  1 !       NONE
;   544          0536  1 !
;   545          0537  1 ! IMPLICIT OUTPUTS:
;   546          0538  1 !
;   547          0539  1 !       NONE
;   548          0540  1 !
;   549          0541  1 ! ROUTINE VALUE:
;   550          0542  1 ! COMPLETION CODES:
;   551          0543  1 !
;   552          0544  1 !       NONE
;   553          0545  1 !
;   554          0546  1 ! SIDE EFFECTS:
;   555          0547  1 !
;   556          0548  1 !       NONE
;   557          0549  1 !
;   558          0550  1 !--
;   559          0551  1
;   560          0552  2     BEGIN
;   561          0553  2
;   562          0554  2     LOCAL
;   563          0555  2         SNK : REF BBLOCK                        ! Local pointer to snk
;   564          0556  2         ;
;   565          0557  2
;   566          0558  2     SNK = .SNKBLK;                              ! Set the pointer
;   567          0559  2     WHILE
;   568          0560  2         .SNK [SNK$L_SRCFL]                      ! Watch for empty queue
;   569          0561  2         NEQ
;   570          0562  2         SNK [SNK$L_SRCFL]
;   571          0563  2     DO
;   572          0564  2         EVL$DEALLOCDBK (.SNK [SNK$L_SRCFL] )    ! Deallocate each source
;   573          0565  2     ;
;   574          0566  2
;   575          0567  2     WHILE                                       ! Watch for end of queue
;   576          0568  2         .SNK [SNK$L_EVTFL]
;   577          0569  2         NEQ
;   578          0570  2         SNK [SNK$L_EVTFL]
;   579          0571  2     DO
;   580          0572  2         EVL$DEALLOCDBK (.SNK [SNK$L_EVTFL] )    ! Deallocate each event
;   581          0573  2     ;
;   582          0574  2
;   583          0575  2     EVL$DEALLOCDBK (.SNK);                      ! Deallocate snk block
;   584          0576  2
```

```
;  585          0577  2     EVL$GB_TRANSDONE = (.EVL$GQ_SNKHEAD EQL EVL$GQ_SNKHEAD); ! Mark done if empty
;  586          0578  2
;  587          0579  1     END;


                                        000C 00000 EVL$DEALLOCSNK:
                                                   .WORD    Save R2,R3                                  ; 0521
                            53      0000V   CF  9E 00002           MOVAB   EVL$DEALLOCDBK, R3          ;
                            52        04    AC  D0 00007           MOVL    SNKBLK, SNK                 ; 0558
                            50        1C    A2  9E 0000B 1$:       MOVAB   28(SNK), R0                 ; 0562
                            50        1C    A2  D1 0000F           CMPL    28(SNK), R0                 ;
                                            08  13 00013           BEQL    2$                          ;
                                      1C    A2  DD 00015           PUSHL   28(SNK)                     ; 0564
                            63              01  FB 00018           CALLS   #1, EVL$DEALLOCDBK          ;
                                            EE  11 0001B           BRB     1$                          ;
                            50        24    A2  9E 0001D 2$:       MOVAB   36(SNK), R0                 ; 0570
                            50        24    A2  D1 00021           CMPL    36(SNK), R0                 ;
                                            08  13 00025           BEQL    3$                          ;
                                      24    A2  DD 00027           PUSHL   36(SNK)                     ; 0572
                            63              01  FB 0002A           CALLS   #1, EVL$DEALLOCDBK          ;
                                            EE  11 0002D           BRB     2$                          ;
                                            52  DD 0002F 3$:       PUSHL   SNK                         ; 0575
                            63              01  FB 00031           CALLS   #1, EVL$DEALLOCDBK          ;
                            51      0000'   CF  9E 00034           MOVAB   EVL$GQ_SNKHEAD, R1          ; 0577
                                            50  D4 00039           CLRL    R0                          ;
                            51      0000'   CF  D1 0003B           CMPL    EVL$GQ_SNKHEAD, R1          ;
                                            02  12 00040           BNEQ    4$                          ;
                                            50  D6 00042           INCL    R0                          ;
                      0000'  CF             50  90 00044 4$:       MOVB    R0, EVL$GB_TRANSDONE        ;
                                            04  00049              RET                                 ; 0579

; Routine Size:  74 bytes,    Routine Base:  $CODE$ + 01F5
```

```
;  589          0580  1 %SBTTL  'EVL$OBTAINEVTS  Obtain Raw Events'
;  590          0581  1 ROUTINE EVL$OBTAINEVTS (ASPBLK) :NOVALUE =
;  591          0582  1
;  592          0583  1 !++
;  593          0584  1 ! FUNCTIONAL DESCRIPTION:
;  594          0585  1 !
;  595          0586  1 !       This routine reads a message from the mailbox associated with the
;  596          0587  1 !       net channel.  We complete the io in the readevts routine and
;  597          0588  1 !       then either try again for another mailbox message, or read
;  598          0589  1 !       events from the NETACP. If no aspblk is available, then we
;  599          0590  1 !       allocate one.
;  600          0591  1 !
;  601          0592  1 ! FORMAL PARAMETERS:
;  602          0593  1 !
;  603          0594  1 !       ASPBLK          Address of aspblk or zero if none
;  604          0595  1 !
;  605          0596  1 ! IMPLICIT INPUTS:
;  606          0597  1 !
;  607          0598  1 !       EVL$W_MBXEVTCHAN
;  608          0599  1 !
;  609          0600  1 ! IMPLICIT OUTPUTS:
;  610          0601  1 !
;  611          0602  1 !       NONE
;  612          0603  1 !
;  613          0604  1 ! ROUTINE VALUE:
;  614          0605  1 ! COMPLETION CODES:
;  615          0606  1 !
;  616          0607  1 !       NONE
;  617          0608  1 !
;  618          0609  1 ! SIDE EFFECTS:
;  619          0610  1 !
;  620          0611  1 !       NONE
;  621          0612  1 !
;  622          0613  1 !--
;  623          0614  1
;  624          0615  2     BEGIN
;  625          0616  2
;  626          0617  2     LOCAL
;  627          0618  2         ASP     : REF BBLOCK,                   ! Ast Paramter block
;  628          0619  2         STATUS
;  629          0620  2         ;
;  630          0621  2
;  631          0622  2     IF .ASPBLK EQL 0
;  632          0623  2     THEN
;  633          0624  2         EVL$ALLOCDBK                            ! Allocate a block if we need
;  634          0625  2             (                                   ! One
;  635          0626  2             EVL$C_EVTBFRSIZE + ASP$C_SIZE,
;  636          0627  2             .EVL$GQ_ASPHEAD [1],
;  637          0628  2             ASP
;  638          0629  2             )
;  639          0630  2     ELSE
;  640          0631  2         ASP = .ASPBLK                           ! Use old block if we have one
;  641          0632  2     ;
;  642          0633  2
;  643          0634  2     ASP [ASP$L_ROUTINE] = EVL$READEVTS;         ! Setup the routine address
;  644          0635  2
;  645        P 0636  2     STATUS = $QIO                               ! Start a read from the
```

```
P 0637  2          (                                        ! mailbox to findout when
P 0638  2          CHAN = .EVL$W_MBXEVTCHAN,                ! events are available
P 0639  2          EFN = EVL$C_ASYNCH_EFN,
P 0640  2          FUNC = IO$_READVBLK,
P 0641  2          IOSB = ASP [ASP$W_IOSB],
P 0642  2          ASTADR = EVL$ASTWORK,
P 0643  2          ASTPRM = .ASP,
P 0644  2          P1 = ASP [ASP$T_DATA],
P 0645  2          P2 = EVL$C_EVTMBXSIZE                    ! Size of mailbox messages
  0646  2          );
  0647  2
  0648  2     EVL$NETERROR (EVL$_OBTEVT, .STATUS, 0)        ! Analyse errors
  0649  2
  0650  1     END;


                                                        .EXTRN  SYS$QIO

                                   0000 00000 EVL$OBTAINEVTS:
                                                        .WORD   Save nothing                    ; 0581
               5E           04  C2 00002            SUBL2   #4, SP
                     04     AC  D5 00005            TSTL    ASPBLK                          ; 0622
                            12  12 00008            BNEQ    1$                              ;
                            5E  DD 0000A            PUSHL   SP                              ; 0625
                  0000'     CF  DD 0000C            PUSHL   EVL$GQ_ASPHEAD+4                ; 0627
               7E 0200      8F  3C 00010            MOVZWL  #512, -(SP)                     ; 0626
        0000V  CF           03  FB 00015            CALLS   #3, EVL$ALLOCDBK                ;
                            04  11 0001A            BRB     2$                              ; 0625
               6E    04     AC  D0 0001C 1$:        MOVL    ASPBLK, ASP                     ; 0631
               50           6E  D0 00020 2$:        MOVL    ASP, R0                         ; 0634
          14   A0  0000V    CF  9E 00023            MOVAB   EVL$READEVTS, 20(R0)            ;
                            7E  7C 00029            CLRQ    -(SP)                           ; 0646
                            7E  7C 0002B            CLRQ    -(SP)                           ;
               7E    40     8F  9A 0002D            MOVZBL  #64, -(SP)                      ;
                     18     A0  9F 00031            PUSHAB  24(R0)                          ;
                            50  DD 00034            PUSHL   R0                              ;
                  0000V     CF  9F 00036            PUSHAB  EVL$ASTWORK                     ;
                     0C     A0  9F 0003A            PUSHAB  12(R0)                          ;
                            31  DD 0003D            PUSHL   #49                             ;
               7E  0000'    CF  3C 0003F            MOVZWL  EVL$W_MBXEVTCHAN, -(SP)         ;
                            02  DD 00044            PUSHL   #2                              ;
    00000000G  00           0C  FB 00046            CALLS   #12, SYS$QIO                    ;
                            7E  D4 0004D            CLRL    -(SP)                           ; 0648
                            50  DD 0004F            PUSHL   STATUS                          ;
              00000000G     8F  DD 00051            PUSHL   #EVL$_OBTEVT                    ;
        0000V  CF           03  FB 00057            CALLS   #3, EVL$NETERROR                ;
                                04 0005C            RET                                     ; 0650

; Routine Size:  93 bytes,    Routine Base:  $CODE$ + 023F
```

```
;  661     0651  1 %SBTTL  'EVL$READEVTS  Read Events from NETACP'
;  662     0652  1 ROUTINE EVL$READEVTS (ASPBLK) :NOVALUE =
;  663     0653  1
;  664     0654  1 !++
;  665     0655  1 ! FUNCTIONAL DESCRIPTION:
;  666     0656  1 !
;  667     0657  1 !       Routine called to obtain a buffer of raw events from NETACP.
;  668     0658  1 !       The input asp has a mailbox message in it which has been read
;  669     0659  1 !       from a net channel mailbox.  The mailbox code is MSG$_EVTAVL for
;  670     0660  1 !       events are available; MSG$_NETSHUT for the network is shutting down.
;  671     0661  1 !       Other codes are ignored.
;  672     0662  1 !       The events are read into an ASP buffer and then schedule work for
;  673     0663  1 !       EVL$PROCESSEVTS.
;  674     0664  1 !
;  675     0665  1 ! FORMAL PARAMETERS:
;  676     0666  1 !
;  677     0667  1 !       ASPBLK          Address of an asp block
;  678     0668  1 !
;  679     0669  1 ! IMPLICIT INPUTS:
;  680     0670  1 !
;  681     0671  1 !       NONE
;  682     0672  1 !
;  683     0673  1 ! IMPLICIT OUTPUTS:
;  684     0674  1 !
;  685     0675  1 !       NONE
;  686     0676  1 !
;  687     0677  1 ! ROUTINE VALUE:
;  688     0678  1 ! COMPLETION CODES:
;  689     0679  1 !
;  690     0680  1 !       NONE
;  691     0681  1 !
;  692     0682  1 ! SIDE EFFECTS:
;  693     0683  1 !
;  694     0684  1 !       NONE
;  695     0685  1 !
;  696     0686  1 !--
;  697     0687  1
;  698     0688  2     BEGIN
;  699     0689  2
;  700     0690  2     LOCAL
;  701     0691  2         ASP     : REF BBLOCK,                   ! Ast Paramter block
;  702     0692  2         NFB     : VECTOR [5, BYTE],             ! Network function block
;  703     0693  2         NFBDSC  : VECTOR [2],                   ! Descriptor of same
;  704     0694  2         RSLDSC  : VECTOR [2],                   ! Descriptor of result buffer
;  705     0695  2         STATUS
;  706     0696  2         ;
;  707     0697  2
;  708     0698  2     ASP = .ASPBLK;                              ! Use old block if we have one
;  709     0699  2
;  710     0700  2     IF NOT EVL$NETERROR                         ! Analyse the error
;  711     0701  2         (
;  712     0702  2         EVL$_OBTEVT,                            ! Unable to obtain events
;  713     0703  2         TRUE,
;  714     0704  2         ASP [ASP$W_IOSB]
;  715     0705  2         )
;  716     0706  2     THEN
;  717     0707  3         BEGIN
```

```
;  718          0708  3          EVL$OBTAINEVTS (.ASP);                  ! Try again
;  719          0709  3          RETURN
;  720          0710  3          END
;  721          0711  2      ;
;  722          0712  2
;  723          0713  2      SELECTONE CH$RCHAR (ASP [ASP$T_DATA] ) OF   ! Dispatch on message types
;  724          0714  2      SET
;  725          0715  2
;  726          0716  2      [MSG$_EVTAVL] :                             ! Just continue with work
;  727          0717  2          0;
;  728          0718  2
;  729          0719  2      [MSG$_NETSHUT] :                            ! Network shut down happening
;  730          0720  3          BEGIN
;  731          0721  3          LOCAL SNKPTR: REF BBLOCK;
;  732          0722  3          SNKPTR = .EVL$GQ_SNKHEAD;
;  733          0723  3          WHILE .SNKPTR NEQ EVL$GQ_SNKHEAD        ! Scan the list
;  734          0724  3          DO
;  735          0725  4              BEGIN
;  736          0726  4              SNKPTR [SNK$V_STS_CLS] = TRUE;      ! Set close as true
;  737          0727  4              WKQ$ADD_WORK_ITEM(EVL$WRITEVT,.SNKPTR); ! Make sure I/O going
;  738          0728  4                                                  ! (will delete when I/O done)
;  739          0729  4              SNKPTR = .SNKPTR [SNK$L_FL];        ! Next snk in list
;  740          0730  3              END;
;  741          0731  3          RETURN;                                 ! Do not reissue mailbox read
;  742          0732  2          END;
;  743          0733  2
;  744          0734  2      [OTHERWISE] :                               ! Ignore the code and
;  745          0735  3          BEGIN
;  746          0736  3          EVL$OBTAINEVTS (.ASP);                  ! try again
;  747          0737  3          RETURN
;  748          0738  2          END;
;  749          0739  2
;  750          0740  2      TES;
;  751          0741  2
;  752          0742  2
;  753          0743  2      ASP [ASP$L_ROUTINE] = EVL$PROCESSEVTS;      ! Setup the routine address
;  754          0744  2
;  755          0745  2      NFBDSC [0] = 5;                             ! Set up the nfb to readevents
;  756          0746  2      NFBDSC [1] = NFB;
;  757          0747  2      NFB [0] = NFB$C_READEVENT;
;  758          0748  2      RSLDSC [0] = EVL$C_EVTBFRSIZE;              ! Set up the result buffer dsc
;  759          0749  2      RSLDSC [1] = ASP [ASP$T_DATA];
;  760        P 0750  2      STATUS = $QIOW                              ! Perform the qio
;  761        P 0751  2          (
;  762        P 0752  2          EFN = EVL$C_SYNCH_EFN,
;  763        P 0753  2          CHAN = .EVL$W_NETEVTCHAN,               ! The channel
;  764        P 0754  2          FUNC = IO$_ACPCONTROL,                  ! function
;  765        P 0755  2          IOSB = ASP [ASP$W_IOSB],
;  766        P 0756  2          P1 = NFBDSC,                            ! Network function block
;  767        P 0757  2          P3 = RSLDSC,                            ! Return result length here
;  768        P 0758  2          P4 = RSLDSC                             ! Result buffer
;  769          0759  2          );
;  770          0760  2
;  771          0761  2      ASP [ASP$W_IOSB1] = .RSLDSC [0];            ! *TEMP* Set length in IOSB
;  772          0762  2
;  773          0763  2      IF NOT .STATUS                              ! Report an error
;  774          0764  2      THEN
```

```
0765 2          SIGNAL_STOP (EVL$_OBTEVT, 0, .STATUS)
0766 2      ;
0767 2
0768 2      WKQ$ADD_WORK_ITEM(EVL$PROCESSEVTS,.ASP);     ! Process the events
0769 2
0770 1      END;



                                   001C 00000 EVL$READEVTS:
                                                     .WORD    Save R2,R3,R4                        ; 0652
             54 00000000G  8F   D0 00002             MOVL     #EVL$_OBTEVT, R4                     ;
                       5E  18   C2 00009             SUBL2    #24, SP                              ;
             53        04  AC   D0 0000C             MOVL     ASPBLK, ASP                          ; 0698
                       0C  A3   9F 00010             PUSHAB   12(ASP)                              ; 0704
                           01   DD 00013             PUSHL    #1                                   ;
                           54   DD 00015             PUSHL    R4                                   ;
      0000V   CF           03   FB 00017             CALLS    #3, EVL$NETERROR                     ;
              31           50   E9 0001C             BLBC     R0, 2$                               ;
              50       18  A3   9A 0001F             MOVZBL   24(ASP), R0                          ; 0713
              3E           50   91 00023             CMPB     R0, #62                              ; 0716
                           30   13 00026             BEQL     3$                                   ;
              3B           50   91 00028             CMPB     R0, #59                              ; 0719
                           23   12 0002B             BNEQ     2$                                   ;
              52    0000'  CF   D0 0002D             MOVL     EVL$GQ_SNKHEAD, SNKPTR               ; 0722
              50    0000'  CF   9E 00032 1$:         MOVAB    EVL$GQ_SNKHEAD, R0                   ; 0723
              50           52   D1 00037             CMPL     SNKPTR, R0                           ;
                           7A   13 0003A             BEQL     5$                                   ;
        2E    A2           10   88 0003C             BISB2    #16, 46(SNKPTR)                      ; 0726
                           52   DD 00040             PUSHL    SNKPTR                               ; 0727
                    0000V  CF   9F 00042             PUSHAB   EVL$WRITEVT                          ;
      0000G   CF           02   FB 00046             CALLS    #2, WKQ$ADD_WORK_ITEM                ;
              52           62   D0 0004B             MOVL     (SNKPTR), SNKPTR                     ; 0729
                           E2   11 0004E             BRB      1$                                   ; 0723
                           53   DD 00050 2$:         PUSHL    ASP                                  ; 0736
      FF4C    CF           01   FB 00052             CALLS    #1, EVL$OBTAINEVTS                   ;
                                04 00057             RET                                           ; 0735
        14    A3    0000V  CF   9E 00058 3$:         MOVAB    EVL$PROCESSEVTS, 20(ASP)             ; 0743
        08    AE           05   D0 0005E             MOVL     #5, NFBDSC                           ; 0745
        0C    AF       10  AE   9E 00062             MOVAB    NFB, NFBDSC+4                        ; 0746
        10    AE           1D   90 00067             MOVB     #29, NFB                             ; 0747
              6E    01E8   8F   3C 0006B             MOVZWL   #488, RSLDSC                         ; 0748
        04    AE       18  A3   9E 00070             MOVAB    24(ASP), RSLDSC+4                    ; 0749
                           7E   7C 00075             CLRQ     -(SP)                                ; 0759
                       08  AE   9F 00077             PUSHAB   RSLDSC                               ;
                       0C  AE   9F 0007A             PUSHAB   RSLDSC                               ;
                           7E   D4 0007D             CLRL     -(SP)                                ;
                       1C  AE   9F 0007F             PUSHAB   NFBDSC                               ;
                           7E   7C 00082             CLRQ     -(SP)                                ;
                       0C  A3   9F 00084             PUSHAB   12(ASP)                              ;
                           38   DD 00087             PUSHL    #56                                  ;
              7E    0000'  CF   3C 00089             MOVZWL   EVL$W_NETEVTCHAN, -(SP)              ;
                           01   DD 0008E             PUSHL    #1                                   ;
  00000000G   00           0C   FB 00090             CALLS    #12, SYS$QIOW                        ;
        0E    A3           6E   B0 00097             MOVW     RSLDSC, 14(ASP)                      ; 0761
```

```
                                 0D              50  E8 0009B          BLBS    STATUS, 4$                                  ; 0763
                                                 50  DD 0009E          PUSHL   STATUS                                      ; 0765
                                                 7E  D4 000A0          CLRL    -(SP)                                       ;
                                                 54  DD 000A2          PUSHL   R4                                          ;
                          00000000G  00          03  FB 000A4          CALLS   #3, LIB$STOP                                ;
                                                 53  DD 000AB 4$:      PUSHL   ASP                                         ; 0768
                                       0000V     CF  9F 000AD          PUSHAB  EVL$PROCESSEVTS                             ;
                               0000G   CF          02  FB 000B1          CALLS   #2, WKQ$ADD_WORK_ITEM                       ;
                                                     04 000B6 5$:      RET                                                 ; 0770

; Routine Size:  183 bytes,    Routine Base:  $CODE$ + 029C
```

```
;  782     0771  1 %SBTTL  'EVL$PROCESSEVTS  Process Raw Events'
;  783     0772  1 ROUTINE EVL$PROCESSEVTS (ASPBLK) :NOVALUE =
;  784     0773  1
;  785     0774  1 !++
;  786     0775  1 ! FUNCTIONAL DESCRIPTION:
;  787     0776  1 !
;  788     0777  1 !       Process raw events and queue them to the various sink nodes.
;  789     0778  1 !       If an error occurred on the read of raw events, signal it and
;  790     0779  1 !       perform another request for raw events.  If no error, scan
;  791     0780  1 !       the raw event buffer and filter each raw event in the buffer.
;  792     0781  1 !
;  793     0782  1 ! FORMAL PARAMETERS:
;  794     0783  1 !
;  795     0784  1 !       ASPBLK          Address of the ASP block of raw events
;  796     0785  1 !
;  797     0786  1 ! IMPLICIT INPUTS:
;  798     0787  1 !
;  799     0788  1 !       EVL$GQ_SNKHEAD  Head of sink list
;  800     0789  1 !
;  801     0790  1 ! IMPLICIT OUTPUTS:
;  802     0791  1 !
;  803     0792  1 !       NONE
;  804     0793  1 !
;  805     0794  1 ! ROUTINE VALUE:
;  806     0795  1 ! COMPLETION CODES:
;  807     0796  1 !
;  808     0797  1 !       NONE
;  809     0798  1 !
;  810     0799  1 ! SIDE EFFECTS:
;  811     0800  1 !
;  812     0801  1 !       NONE
;  813     0802  1 !
;  814     0803  1 !--
;  815     0804  1
;  816     0805  2     BEGIN
;  817     0806  2
;  818     0807  2     LOCAL
;  819     0808  2         ASP : REF BBLOCK,               ! Pointers to various blocks we need
;  820     0809  2         RAW : REF BBLOCK,
;  821     0810  2         SNK : REF BBLOCK,
;  822     0811  2         RAWPTR,                         ! Pointer to a raw event
;  823     0812  2         RAWEND,                         ! End of all of them
;  824     0813  2         RAWDSC : VECTOR [2]             ! Descriptor of raw event
;  825     0814  2         ;
;  826     0815  2
;  827     0816  2
;  828     0817  2     ASP = .ASPBLK;                      ! If some error then,
;  829     0818  2     IF NOT EVL$NETERROR (EVL$_OBTEVT, TRUE, ASP [ASP$W_IOSB] )
;  830     0819  2     THEN
;  831     0820  3         BEGIN
;  832     0821  3         EVL$OBTAINEVTS (.ASP);          ! Just try again for some events
;  833     0822  3         RETURN
;  834     0823  3         END
;  835     0824  2     ;
;  836     0825  2
;  837     0826  2     RAWPTR = ASP [ASP$T_DATA];          ! Set start pointer and
;  838     0827  2     RAWEND = .RAWPTR + .ASP [ASP$W_IOSB1]; ! end pointer
```

```
;  839     0828  2
;  840     0829  2      WHILE .RAWPTR LSSA .RAWEND          ! Scan the whole list of raw events
;  841     0830  2      DO
;  842     0831  2          BEGIN
;  843     0832  2          RAW = .RAWPTR;                  ! Point to this event
;  844     0833  2          RAWPTR = .RAWPTR + .RAW [RAW$W_BYTES]; ! and the next one
;  845     0834  2          IF .RAWPTR GTRA .RAWEND         ! Is this event too long??
;  846     0835  2          THEN
;  847     0836  4              BEGIN
;  848     0837  4              SIGNAL (EVL$_RAWFMT);       ! Signal some error and
;  849     0838  4              EXITLOOP                    ! then stop
;  850     0839  4              END
;  851     0840  2          ;
;  852     0841  2
;  853     0842  2          RAWDSC [0] = .RAW [RAW$W_BYTES]; ! Build descriptor of raw event
;  854     0843  2          RAWDSC [1] = .RAW;
;  855     0844  2          EVL$PRINTLOG                    ! Log the raw event if required
;  856     0845  2              (
;  857     0846  2              $BITPOSITION (ELG$V_RAWEVT), ! Bit number of responsible bit
;  858     0847  2              %ASCID 'Raw event',         ! Header text
;  859     0848  2              0,                          ! No extra text
;  860     0849  2              RAWDSC                      ! Descriptor of data
;  861     0850  2              );
;  862     0851  2
;  863     0852  2          IF  .RAW [RAW$W_EVTCODE]        ! Data base change event?
;  864     0853  2              EQL
;  865     0854  2              EVC$C_VMS_DBC
;  866     0855  2          THEN
;  867     0856  3              EVL$BLDFILTERS ()           ! Just rebuild the filters
;  868     0857  3          ELSE
;  869     0858  4              BEGIN
;  870     0859  4              SNK = .EVL$GQ_SNKHEAD [0];  ! Scan the sink queue and
;  871     0860  4              WHILE .SNK NEQA EVL$GQ_SNKHEAD
;  872     0861  4              DO
;  873     0862  5                  BEGIN
;  874     0863  5                  EVL$FILTEREVT (.RAW, .SNK);     ! filter each raw event
;  875     0864  5                  SNK = .SNK [SNK$L_FL]   ! look at next sink
;  876     0865  5                  END
;  877     0866  4              END
;  878     0867  3          END
;  879     0868  2      ;
;  880     0869  2
;  881     0870  2      EVL$OBTAINEVTS (.ASP)               ! Back for some raw events
;  882     0871  2
;  883     0872  1      END;


                                                            .PSECT  $PLIT$,NOWRT,NOEXE,2

            00  00  00  74  6E  65  76  65  20  77  61  52  00048 P.AAI:  .ASCII  \Raw event\<0><0><0>                ;
                                                        010E0009  00054 P.AAH:  .LONG   17694729                     ;
                                                        00000000' 00058        .ADDRESS P.AAI                       ;


                                                            .PSECT  $CODE$,NOWRT,2
```

```
                                 007C 00000 EVL$PROCESSEVTS:
                                                .WORD   Save R2,R3,R4,R5,R6                     ; 0772
                          5E                    08  C2 00002         SUBL2   #8, SP
                          53          04        AC  D0 00005         MOVL    ASPBLK, ASP           ; 0817
                                      0C        A3  9F 00009         PUSHAB  12(ASP)               ; 0818
                                                01  DD 0000C         PUSHL   #1
                                00000000G       8F  DD 0000E         PUSHL   #EVL$_OBTEVT
                      0000V     CF              03  FB 00014         CALLS   #3, EVL$NETERROR
                          6D                    50  E9 00019         BLBC    R0, 5$
                          52          18        A3  9E 0001C         MOVAB   24(R3), RAWPTR        ; 0826
                          56          0E        A3  3C 00020         MOVZWL  14(ASP), RAWEND       ; 0827
                          56                    52  C0 00024         ADDL2   RAWPTR, RAWEND
                          56                    52  D1 00027 1$:     CMPL    RAWPTR, RAWEND        ; 0829
                                                5D  1E 0002A         BGEQU   5$
                          54                    52  D0 0002C         MOVL    RAWPTR, RAW           ; 0832
                          50                    64  3C 0002F         MOVZWL  (RAW), R0             ; 0833
                          52                    50  C0 00032         ADDL2   R0, RAWPTR
                          56                    52  D1 00035         CMPL    RAWPTR, RAWEND        ; 0834
                                                0F  1B 00038         BLEQU   2$
                                00000000G       8F  DD 0003A         PUSHL   #EVL$_RAWFMT          ; 0837
                00000000G     00                01  FB 00040         CALLS   #1, LIB$SIGNAL
                                                40  11 00047         BRB     5$                    ; 0836
                          6E                    64  3C 00049 2$:     MOVZWL  (RAW), RAWDSC         ; 0842
                            04  AE              54  D0 0004C         MOVL    RAW, RAWDSC+4         ; 0843
                                                5E  DD 00050         PUSHL   SP                    ; 0845
                                                7E  D4 00052         CLRL    -(SP)
                                      0000'     CF  9F 00054         PUSHAB  P.AAH                 ; 0846
                                                04  DD 00058         PUSHL   #4                    ; 0845
                          0000G     CF              04  FB 0005A         CALLS   #4, EVL$PRINTLOG
                          2000      8F        0A    A4  B1 0005F         CMPW    10(RAW), #8192        ; 0853
                                                07  12 00065         BNEQ    3$
                          FCC4      CF              00  FB 00067         CALLS   #0, EVL$BLDFILTERS    ; 0856
                                                B9  11 0006C         BRB     1$
                                55        0000'   CF  D0 0006E 3$:     MOVL    EVL$GQ_SNKHEAD, SNK   ; 0859
                                50        0000'   CF  9E 00073 4$:     MOVAB   EVL$GQ_SNKHEAD, R0    ; 0860
                                50                55  D1 00078         CMPL    SNK, R0
                                                AA  13 0007B         BEQL    1$
                                                30  BB 0007D         PUSHR   #^M<R4,R5>            ; 0863
                          0000V     CF              02  FB 0007F         CALLS   #2, EVL$FILTEREVT
                                55                65  D0 00084         MOVL    (SNK), SNK            ; 0864
                                                EA  11 00087         BRB     4$
                                                53  DD 00089 5$:     PUSHL   ASP                   ; 0870
                          FE5C      CF              01  FB 0008B         CALLS   #1, EVL$OBTAINEVTS
                                                    04  00090         RET                           ; 0872

; Routine Size:  145 bytes,    Routine Base:  $CODE$ + 0353
```

```
;  885          0873  1 %SBTTL  'EVL$FILTEREVT  Filter Events with a Sink'
;  886          0874  1 ROUTINE EVL$FILTEREVT (RAWEVT, SNKBLK) :NOVALUE =
;  887          0875  1
;  888          0876  1 !++
;  889          0877  1 ! FUNCTIONAL DESCRIPTION:
;  890          0878  1 !
;  891          0879  1 !       Filter a single event with a single sink.  If the event matches
;  892          0880  1 !       for any of the sources or the global filters, then its translated and
;  893          0881  1 !       queued for this sink.
;  894          0882  1 !
;  895          0883  1 ! FORMAL PARAMETERS:
;  896          0884  1 !
;  897          0885  1 !       RAWEVT          Address of the raw event in the buffer
;  898          0886  1 !       SNKBLK          Address of the sink control block
;  899          0887  1 !
;  900          0888  1 ! IMPLICIT INPUTS:
;  901          0889  1 !
;  902          0890  1 !       NONE
;  903          0891  1 !
;  904          0892  1 ! IMPLICIT OUTPUTS:
;  905          0893  1 !
;  906          0894  1 !       NONE
;  907          0895  1 !
;  908          0896  1 ! ROUTINE VALUE:
;  909          0897  1 ! COMPLETION CODES:
;  910          0898  1 !
;  911          0899  1 !       NONE
;  912          0900  1 !
;  913          0901  1 ! SIDE EFFECTS:
;  914          0902  1 !
;  915          0903  1 !       NONE
;  916          0904  1 !
;  917          0905  1 !--
;  918          0906  1
;  919          0907  2     BEGIN
;  920          0908  2
;  921          0909  2     LOCAL
;  922          0910  2         RAW : REF BBLOCK,               ! Pointer to a raw event
;  923          0911  2         SNK : REF BBLOCK,               ! Pointer to the sink block
;  924          0912  2         SRC : REF BBLOCK,               ! Pointer to a source descriptor .
;  925          0913  2         SINKMASK : BYTE,                ! Sink mask (console, file, monitor)
;  926          0914  2         SRCFOUND : BYTE                 ! Bool for source found
;  927          0915  2         ;
;  928          0916  2
;  929          0917  2     RAW = .RAWEVT;                          ! Set pointer for raw event
;  930          0918  2     SNK = .SNKBLK;                          ! Set pointer to sink block
;  931          0919  2     SINKMASK = 0;                           ! No sinks indicated yet
;  932          0920  2     SRCFOUND = FALSE;                       ! Specific filter not found
;  933          0921  2
;  934          0922  2     SRC = .SNK [SNK$L_SRCFL];               ! Scan the source dsc from
;  935          0923  2     WHILE .SRC NEQ SNK [SNK$L_SRCFL]        ! the head of the list
;  936          0924  2     DO
;  937          0925  3         BEGIN
;  938          0926  3         IF   .RAW [RAW$B_SRCTYP]            ! If source types match
;  939          0927  3              EQL
;  940          0928  3              .SRC [SRC$B_SRCTYP]
;  941          0929  3              AND
```

```
;  942     0930  4            BEGIN                                   ! and the sources themselves
;  943     0931  4            SELECTONEU .RAW [RAW$B_SRCTYP] OF       ! match according to the type
;  944     0932  4            SET
;  945     0933  4            [EVC$C_SRC_NON] : TRUE ;                ! No source type, always match
;  946     0934  4            [EVC$C_SRC_LIN,EVC$C_SRC_CIR,           ! Compare ASCIC strings
;  947     0935  4                EVC$C_SRC_MOD] :
;  948     0936  5                BEGIN
;  949     0937  5                CH$EQL
;  950     0938  5                    (
;  951     0939  5                    CH$RCHAR (SRC [SRC$T_SRCID] ), ! Line in the source block
;  952     0940  5                    SRC [SRC$T_SRCID] + 1,         ! size and address
;  953     0941  5                    CH$RCHAR (RAW [RAW$T_SRCID]),  ! Counted string
;  954     0942  5                    RAW [RAW$T_SRCID] + 1,         ! from the raw event
;  955     0943  5                    0                              ! fill
;  956     0944  5                    )
;  957     0945  4                END;
;  958     0946  4            [EVC$C_SRC_NOD] :                       ! Compare node id's
;  959     0947  5                BEGIN
;  960     0948  5                .( RAW [RAW$T_SRCID] ) <0, 16>      ! Compare only the node
;  961     0949  5                EQL                                 ! addresses
;  962     0950  5                .( SRC [SRC$T_SRCID] ) <0, 16>
;  963     0951  4                END;
;  964     0952  4
;  965     0953  4            [EVC$C_SRC_ARE] :                       ! Compare area numbers
;  966     0954  4                (CH$RCHAR(RAW [RAW$T_SRCID]) EQL CH$RCHAR(SRC [SRC$T_SRCID]));
;  967     0955  4            TES
;  968     0956  4            END
;  969     0957  3        THEN
;  970     0958  4            BEGIN
;  971     0959  4            SRCFOUND = TRUE;                        ! They match, so we found it
;  972     0960  4            SINKMASK = .SINKMASK OR                 ! Add its sink contribution
;  973     0961  4                EVL$FILTERSRC (.RAW, .SRC)          ! to the growing mask
;  974     0962  4            END
;  975     0963  3        ;
;  976     0964  3            SRC = .SRC [SRC$L_FL]                   ! Link to the next source dsc
;  977     0965  3            END
;  978     0966  2    ;
;  979     0967  2
;  980     0968  2    IF NOT .SRCFOUND                                ! If we did not find a source
;  981     0969  2    THEN
;  982     0970  3        BEGIN
;  983     0971  3        SRC = .SNK [SNK$L_SRCFL];                   ! Scan the source list
;  984     0972  3        WHILE .SRC NEQ SNK [SNK$L_SRCFL]            ! and look for global filters
;  985     0973  3        DO
;  986     0974  4            BEGIN
;  987     0975  4            IF  .SRC [SRC$B_SRCTYP]                 ! Consider only global filters
;  988     0976  4                EQL
;  989     0977  4                EVC$C_SRC_NON
;  990     0978  4            THEN
;  991     0979  5                BEGIN
;  992     0980  5                SINKMASK = .SINKMASK OR             ! Combine its contribution
;  993     0981  5                    EVL$FILTERSRC (.RAW, .SRC)
;  994     0982  5                END
;  995     0983  4            ;
;  996     0984  4            SRC = .SRC [SRC$L_FL]                   ! And look at remainder of
;  997     0985  4            END                                     ! source descriptors
;  998     0986  3        END
```

```
;  999     0987  2     ;
; 1000     0988  2
; 1001     0989  2     IF .SINKMASK NEQ 0                          ! If we have any sinks
; 1002     0990  2     THEN
; 1003     0991  2         EVL$QUEUEVT (.RAW, .SNK, .SINKMASK)     ! Queue the event to the sink
; 1004     0992  2
; 1005     0993  1     END;


                                    01FC 00000 EVL$FILTEREVT:
                                                       .WORD   Save R2,R3,R4,R5,R6,R7,R8                    ; 0874
                          55    04  AC  7D 00002             MOVQ    RAWEVT, RAW                          ; 0917
                                    58  94 00006             CLRB    SINKMASK                             ; 0919
                                    57  94 00008             CLRB    SRCFOUND                             ; 0920
                          54    1C  A6  D0 0000A             MOVL    28(SNK), SRC                         ; 0922
                          50    1C  A6  9E 0000E 1$:         MOVAB   28(SNK), R0                          ; 0923
                          50        54  D1 00012             CMPL    SRC, R0                              ;
                                    5D  13 00015             BEQL    8$                                   ;
                       0B A4    0C  A5  91 00017             CMPB    12(RAW), 11(SRC)                     ; 0928
                                    51  12 0001C             BNEQ    7$                                   ;
                          50    0C  A5  9A 0001E             MOVZBL  12(RAW), R0                          ; 0931
                       FF 8F        50  91 00022             CMPB    R0, #255                             ; 0933
                                    38  13 00026             BEQL    6$                                   ;
                          01        50  91 00028             CMPB    R0, #1                               ; 0934
                                    0A  13 0002B             BEQL    2$                                   ;
                          03        50  91 0002D             CMPB    R0, #3                               ;
                                    17  1F 00030             BLSSU   3$                                   ;
                          04        50  91 00032             CMPB    R0, #4                               ;
                                    12  1A 00035             BGTRU   3$                                   ;
                          51    0C  A4  9A 00037 2$:         MOVZBL  12(SRC), R1                          ; 0939
                          50    0D  A5  9A 0003B             MOVZBL  13(RAW), R0                          ; 0941
                50        00    0D  A4  51  2D 0003F         CMPC5   R1, 13(SRC), #0, R0, 14(RAW)         ;
                                    0E  A5     00045         ;
                                    15  11 00047             BRB     5$                                   ;
                                    50  D5 00049 3$:         TSTL    R0                                   ; 0946
                                    07  12 0004B             BNEQ    4$                                   ;
                             0C A4  0D  A5  B1 0004D         CMPW    13(RAW), 12(SRC)                     ; 0950
                                    0A  11 00052             BRB     5$                                   ;
                          05        50  91 00054 4$:         CMPB    R0, #5                               ; 0953
                                    07  12 00057             BNEQ    6$                                   ;
                             0C A4  0D  A5  91 00059         CMPB    13(RAW), 12(SRC)                     ; 0954
                                    0F  12 0005E 5$:         BNEQ    7$                                   ;
                          57        01  90 00060 6$:         MOVB    #1, SRCFOUND                         ; 0959
                                    54  DD 00063             PUSHL   SRC                                  ; 0961
                                    55  DD 00065             PUSHL   RAW                                  ;
                       0000V CF         02  FB 00067         CALLS   #2, EVL$FILTERSRC                    ;
                          58            50  88 0006C         BISB2   R0, SINKMASK                         ;
                          54            64  D0 0006F 7$:     MOVL    (SRC), SRC                           ; 0964
                                        9A  11 00072         BRB     1$                                   ;
                          25            57  E8 00074 8$:     BLBS    SRCFOUND, 11$                        ; 0968
                          54    1C      A6  D0 00077         MOVL    28(SNK), SRC                         ; 0971
                          50    1C      A6  9E 0007B 9$:     MOVAB   28(SNK), R0                          ; 0972
                          50            54  D1 0007F         CMPL    SRC, R0                              ;
                                        18  13 00082         BEQL    11$                                  ;
```

```
                          FF   8F        0B    A4  91 00084          CMPB    11(SRC), #255                                   ; 0976
                                             0C  12 00089          BNEQ    10$
                                             54  DD 0008B          PUSHL   SRC                                             ; 0981
                                             55  DD 0008D          PUSHL   RAW
                        0000V CF             02  FB 0008F          CALLS   #2, EVL$FILTERSRC
                              58             50  88 00094          BISB2   R0, SINKMASK
                              54             64  D0 00097 10$:     MOVL    (SRC), SRC                                      ; 0984
                                             DF  11 0009A          BRB     9$
                                             58  95 0009C 11$:     TSTB    SINKMASK                                        ; 0989
                                             0B  13 0009E          BEQL    12$
                              7E             58  9A 000A0          MOVZBL  SINKMASK, -(SP)                                 ; 0991
                              7E             55  7D 000A3          MOVQ    RAW, -(SP)
                        0000V CF             03  FB 000A6          CALLS   #3, EVL$QUEUEVT
                                                 04 000AB 12$:     RET                                                     ; 0993

; Routine Size:  172 bytes,    Routine Base:  $CODE$ + 03E4
```

```
;1007     0994  1 %SBTTL  'EVL$FILTERSRC  Filter an Event with one Source'
;1008     0995  1 ROUTINE EVL$FILTERSRC (RAWEVT, SRCBLK) =
;1009     0996  1
;1010     0997  1 !++
;1011     0998  1 ! FUNCTIONAL DESCRIPTION:
;1012     0999  1 !
;1013     1000  1 !       Scan the filters with one source block and if the raw event is
;1014     1001  1 !       passed by the filters, return the sink mask value as the value
;1015     1002  1 !       of the routine.
;1016     1003  1 !
;1017     1004  1 ! FORMAL PARAMETERS:
;1018     1005  1 !
;1019     1006  1 !       RAWEVT          Address of the raw event
;1020     1007  1 !       SRCBLK          Address of the source descriptor block
;1021     1008  1 !
;1022     1009  1 ! IMPLICIT INPUTS:
;1023     1010  1 !
;1024     1011  1 !       NONE
;1025     1012  1 !
;1026     1013  1 ! IMPLICIT OUTPUTS:
;1027     1014  1 !
;1028     1015  1 !       NONE
;1029     1016  1 !
;1030     1017  1 ! ROUTINE VALUE:
;1031     1018  1 ! COMPLETION CODES:
;1032     1019  1 !
;1033     1020  1 !       If the event is passed by the filters, return the sinkmask from the
;1034     1021  1 !       source.  Otherwise return zero.
;1035     1022  1 !
;1036     1023  1 ! SIDE EFFECTS:
;1037     1024  1 !
;1038     1025  1 !       NONE
;1039     1026  1 !
;1040     1027  1 !--
;1041     1028  1
;1042     1029  2     BEGIN
;1043     1030  2
;1044     1031  2     LOCAL
;1045     1032  2         RAW     : REF BBLOCK,                   ! Pointer to raw event
;1046     1033  2         SRC     : REF BBLOCK,                   ! Pointer to source descriptor
;1047     1034  2         FLT     : REF BBLOCK,                   ! Pointer to filter
;1048     1035  2         CLASS,                                  ! Class from raw event
;1049     1036  2         TYPE                                    ! Type number from raw event
;1050     1037  2         ;
;1051     1038  2
;1052     1039  2     RAW = .RAWEVT;                              ! Setup useful pointers
;1053     1040  2     SRC = .SRCBLK;
;1054     1041  2     FLT = SRC [SRC$T_FILTERS];                  ! First filter is here
;1055     1042  2     CLASS = .RAW [RAW$V_EVTCLS];                ! Event code from raw event
;1056     1043  2     TYPE = .RAW [RAW$V_EVTTYP];                 ! Obtain type number
;1057     1044  2
;1058     1045  2     DECRU FLTS FROM .SRC [SRC$W_FILTERS] TO 1   ! Look at all the filters
;1059     1046  2     DO
;1060     1047  3         BEGIN
;1061     1048  3         IF                                      ! Compare according to filter
;1062     1049  4             BEGIN                               ! class and wild codes
;1063     1050  4             SELECTONE .FLT [FLT$V_WLDCOD] OF
```

```
1051 4                   SET
1052 4                   [EVC$C_WLDCLS_KNO] :                 ! Known events
1053 4                       TRUE;
1054 4                   [EVC$C_WLDCLS_ALL] :                 ! All types of a class
1055 5                       BEGIN
1056 5                       IF .CLASS EQL .FLT [FLT$V_CLASS]
1057 5                       THEN TRUE
1058 5                       ELSE FALSE
1059 4                       END;
1060 4                   [0] :                                ! Class and type must match
1061 5                       BEGIN
1062 5                       IF .CLASS EQL .FLT [FLT$V_CLASS]
1063 5                           AND                          ! Check both type masks
1064 5                           .BBLOCK [FLT [FLT$Q_TYPESLOG], 0, .TYPE, 1, 0]
1065 5                           AND
1066 5                           NOT .BBLOCK [FLT [FLT$Q_TYPESFIL], 0, .TYPE, 1, 0]
1067 5                       THEN TRUE
1068 5                       ELSE FALSE
1069 4                       END;
1070 4                   TES
1071 4                   END
1072 3               THEN RETURN 1^(.SRC [SRC$B_SNKTYPE]-1); ! Return the sink mask
1073 3               FLT = .FLT + FLT$C_SIZE              ! Next filter
1074 3               END
1075 2           ;
1076 2           RETURN 0                                 ! No sink found
1077 2
1078 1           END;



                                      003C 00000 EVL$FILTERSRC:
                                                        .WORD   Save R2,R3,R4,R5                      ; 0995
                        52      04  AC  D0 00002        MOVL    RAWEVT, RAW                           ; 1039
                        51      08  AC  D0 00006        MOVL    SRCBLK, SRC                           ; 1040
                        50      20  A1  9E 0000A        MOVAB   32(R1), FLT                           ; 1041
 55      0A  A2         09          06  EF 0000E        EXTZV   #6, #9, 10(RAW), CLASS                ; 1042
 54      0A  A2         06          00  EF 00014        EXTZV   #0, #6, 10(RAW), TYPE                 ; 1043
                        53      1E  A1  3C 0001A        MOVZWL  30(SRC), FLTS                         ; 1045
                                    42  11 0001E        BRB     5$                                    ;
 03          60         02          0E  ED 00020 1$:    CMPZV   #14, #2, (FLT), #3                    ; 1052
                                    28  13 00025        BEQL    3$                                    ;
 02          60         02          0E  ED 00027        CMPZV   #14, #2, (FLT), #2                    ; 1054
                                    09  12 0002C        BNEQ    2$                                    ;
 55          60         09          00  ED 0002E        CMPZV   #0, #9, (FLT), CLASS                  ; 1056
                                    28  12 00033        BNEQ    4$                                    ;
                                    18  11 00035        BRB     3$                                    ;
             C0     8F  01          A0  93 00037 2$:    BITB    1(FLT), #192                          ; 1060
                                    11  12 0003C        BNEQ    3$                                    ;
 55          60         09          00  ED 0003E        CMPZV   #0, #9, (FLT), CLASS                  ; 1062
                                    18  12 00043        BNEQ    4$                                    ;
             13     04  A0          54  E1 00045        BBC     TYPE, 4(FLT), 4$                      ; 1064
             0E     0C  A0          54  E0 0004A        BBS     TYPE, 12(FLT), 4$                     ; 1066
                        52      0A  A1  9A 0004F 3$:    MOVZBL  10(SRC), R2                           ; 1072
                                    52  D7 00053        DECL    R2                                    ;
```

```
                52              01              52  78 00055          ASHL    R2, #1, R2                                      ;
                                50              52  D0 00059          MOVL    R2, R0                                          ;
                                                    04 0005C          RET                                                     ;
                                50              14  C0 0005D 4$:      ADDL2   #20, FLT                                        ; 1073
                                                53  D7 00060          DECL    FLT$                                            ;
                                                BC  12 00062 5$:      BNEQ    1$                                              ;
                                                50  D4 00064          CLRL    R0                                              ; 1076
                                                    04 00066          RET                                                     ; 1078

; Routine Size:  103 bytes,    Routine Base:  $CODE$ + 0490
```

```
;  1093          1079  1 %SBTTL  'EVL$QUEUEVT  Translate and Queue an Event'
;  1094          1080  1 ROUTINE EVL$QUEUEVT (RAWEVT, SNKBLK, SNKMASK) :NOVALUE =
;  1095          1081  1
;  1096          1082  1 !++
;  1097          1083  1 ! FUNCTIONAL DESCRIPTION:
;  1098          1084  1 !
;  1099          1085  1 !       Translate the raw event to DNA standard form, queue it to the
;  1100          1086  1 !       sink block and call the EVL$WRITEVT routine to start the
;  1101          1087  1 !       output process to the sink node.
;  1102          1088  1 !
;  1103          1089  1 ! FORMAL PARAMETERS:
;  1104          1090  1 !
;  1105          1091  1 !       RAWEVT          Address of the raw event
;  1106          1092  1 !       SNKBLK          Address of the sink block
;  1107          1093  1 !       SNKMASK         Value of the sink mask to use
;  1108          1094  1 !
;  1109          1095  1 ! IMPLICIT INPUTS:
;  1110          1096  1 !
;  1111          1097  1 !       NONE
;  1112          1098  1 !
;  1113          1099  1 ! IMPLICIT OUTPUTS:
;  1114          1100  1 !
;  1115          1101  1 !       NONE
;  1116          1102  1 !
;  1117          1103  1 ! ROUTINE VALUE:
;  1118          1104  1 ! COMPLETION CODES:
;  1119          1105  1 !
;  1120          1106  1 !       NONE
;  1121          1107  1 !
;  1122          1108  1 ! SIDE EFFECTS:
;  1123          1109  1 !
;  1124          1110  1 !       NONE
;  1125          1111  1 !
;  1126          1112  1 !--
;  1127          1113  1
;  1128          1114  2     BEGIN
;  1129          1115  2
;  1130          1116  2     LITERAL
;  1131          1117  2         EVTSIZE = 300                       ! Size of temp event buffer
;  1132          1118  2         ;
;  1133          1119  2
;  1134          1120  2     LOCAL
;  1135          1121  2         EVTBFR  : VECTOR [EVTSIZE, BYTE],   ! Place to build DNA event
;  1136          1122  2         RAW     : REF BBLOCK,               ! Pointer to raw event
;  1137          1123  2         SNK     : REF BBLOCK,               ! Pointer to sink block
;  1138          1124  2         EVQ     : REF BBLOCK,               ! Pointer to event block
;  1139          1125  2         PTR,                                ! Pointer to build event
;  1140          1126  2         PTRADR,                             ! Pointer in memory for parm
;  1141          1127  2         SIZE,                               ! Size of event
;  1142          1128  2         EVTDSC  : VECTOR [2],               ! Descriptor of event
;  1143          1129  2         DATAPTR,                            ! Pointer to data of event
;  1144          1130  2         HALFDAY,                            ! Julian half days
;  1145          1131  2         SECS,                               ! Seconds in half day
;  1146          1132  2         MSECS                               ! Milliseconds in second
;  1147          1133  2         ;
;  1148          1134  2
;  1149          1135  2     RAW = .RAWEVT;                          ! Set pointer to raw event
```

```
;  1150     1136  2      PTR = EVTBFR;                                   ! Pointer to build DNA event
;  1151     1137  2      SNK = .SNKBLK;                                  ! Point to the sink
;  1152     1138  2
;  1153     1139  2      IF .SNK [SNK$W_EVTCNT] GTRU EVL$C_MAXEVTCNT ! Event queue too long?
;  1154     1140  2      THEN
;  1155     1141  3          BEGIN
;  1156     1142  4          IF (                                        ! If sink has not been lost
;  1157     1143  4              NOT .SNK [SNK$B_SNKLOS]                 ! of this type yet
;  1158     1144  4              AND .SNKMASK
;  1159     1145  4              )
;  1160     1146  3              NEQ 0
;  1161     1147  3          THEN
;  1162     1148  4              BEGIN
;  1163     1149  4              EVL$BLDLOSTEVENT                        ! Build lost event message
;  1164     1150  4                  (                                   ! For the correct sinks
;  1165     1151  4                  .SNKMASK AND NOT .SNK [SNK$B_SNKLOS],
;  1166     1152  4                  .SNK [SNK$L_EVTBL],                 ! Link in at tail of queue
;  1167     1153  4                  EVQ                                 ! Return address here
;  1168     1154  4                  );
;  1169     1155  4              SNK [SNK$B_SNKLOS] =                    ! Remember the sinks we lost
;  1170     1156  4                  .SNK [SNK$B_SNKLOS] OR .SNKMASK; ! So only one message
;  1171     1157  4              SNK [SNK$W_EVTCNT] =                    ! Count another event
;  1172     1158  4                  .SNK [SNK$W_EVTCNT] + 1;
;  1173     1159  4              EVTDSC [0] = .EVQ [EVQ$W_EVTSIZE];  ! Build descriptor of event
;  1174     1160  4              EVTDSC [1] = EVQ [EVQ$T_EVENT];
;  1175     1161  4              EVL$PRINTLOG                            ! Log event if requested
;  1176     1162  4                  (
;  1177     1163  4                  $BITPOSITION (ELG$V_QUEEVT),
;  1178     1164  4                  %ASCID 'Events Lost',
;  1179     1165  4                  SNK [SNK$L_SNKLEN],
;  1180     1166  4                  EVTDSC
;  1181     1167  4                  )
;  1182     1168  4              END
;  1183     1169  3          ;
;  1184     1170  3          RETURN                                      ! We lost the event, so done
;  1185     1171  3          END
;  1186     1172  2      ELSE
;  1187     1173  2          SNK [SNK$B_SNKLOS] = FALSE                  ! No events lost now
;  1188     1174  2      ;
;  1189     1175  2
;  1190     1176  2
;  1191     1177  2      IF .RAW [RAW$W_BYTES]                           ! Quick check of raw size
;  1192     1178  3              GTRU EVTSIZE - (-27+37)                 ! to see if it fits in buffer
;  1193     1179  2          OR .RAW [RAW$W_BYTES]                       ! Or if too small
;  1194     1180  2              LSSU $BYTEOFFSET(RAW$T_DATA)
;  1195     1181  2      THEN
;  1196     1182  3          BEGIN
;  1197     1183  3          SIGNAL (EVL$_EVTSIZ);                       ! Signal some thing bad wrong
;  1198     1184  3          RETURN                                      ! and ignore event
;  1199     1185  3          END
;  1200     1186  2      ;
;  1201     1187  2
;  1202     1188  2      EVL$JULIAN                                      ! Convert raw time to DNA
;  1203     1189  2          (                                           ! format
;  1204     1190  2          RAW [RAW$T_SYSTIM],                         ! 64 bit systime
;  1205     1191  2          HALFDAY, SECS, MSECS                        ! Julian halfdays, seconds, ..
;  1206     1192  2          );
```

```
;  1207     1193  2
;  1208     1194  2 !
;  1209     1195  2 !       Build the DNA format event in the buffer
;  1210     1196  2 !
;  1211     1197  2
;  1212     1198  2     CH$WCHAR_A (1, PTR);                            ! Just a code byte for event
;  1213     1199  2     CH$WCHAR_A (.SNKMASK, PTR);                     ! The sink mask byte
;  1214     1200  2     PTR = CH$COPY                                   ! Build more of the header
;  1215     1201  2         (
;  1216     1202  2         2, RAW [RAW$W_EVTCODE],                     ! Event code
;  1217     1203  2         2, HALFDAY,                                 ! Date and time in dna format
;  1218     1204  2         2, SECS,
;  1219     1205  2         2, MSECS,
;  1220     1206  2         0,                                          ! Just a dummy fill char
;  1221     1207  2         8, .PTR                                     ! Place to put it
;  1222     1208  2         );
;  1223     1209  2
;  1224     1210  2     PTR = CH$MOVE                                   ! Put in the source node
;  1225     1211  2         (                                           ! in node id format
;  1226     1212  2         .EVL$GB_LOCALNODE,                          ! from a special buffer
;  1227     1213  2         EVL$GT_LOCALNODE,                           ! Built in advance
;  1228     1214  2         .PTR
;  1229     1215  2         );
;  1230     1216  2
;  1231     1217  2     CH$WCHAR_A (.RAW [RAW$B_SRCTYP], PTR);          ! Copy the source type
;  1232     1218  2
;  1233     1219  2     SELECTONEU .RAW [RAW$B_SRCTYP] OF               ! and id depending on type
;  1234     1220  2     SET
;  1235     1221  2     [EVC$C_SRC_NON] : (0);                          ! Nothing for no source id
;  1236     1222  2     [EVC$C_SRC_LIN,EVC$C_SRC_CIR,                   ! Copy ASCIC string
;  1237     1223  2         EVC$C_SRC_MOD] :
;  1238     1224  3         BEGIN
;  1239     1225  3         PTR = CH$MOVE                               ! Line id is a counted string
;  1240     1226  3             (
;  1241     1227  3             CH$RCHAR (RAW [RAW$T_SRCID]) + 1,
;  1242     1228  3             RAW [RAW$T_SRCID],
;  1243     1229  3             .PTR
;  1244     1230  3             )
;  1245     1231  2         END;
;  1246     1232  2     [EVC$C_SRC_NOD] :                               ! Copy node id from raw
;  1247     1233  3         BEGIN                                       ! its word (adr) ascic (name)
;  1248     1234  3         PTR = CH$MOVE
;  1249     1235  3             (
;  1250     1236  3             CH$RCHAR (RAW [RAW$T_SRCID] + 2) + 3, ! Count and address with str
;  1251     1237  3             RAW [RAW$T_SRCID],
;  1252     1238  3             .PTR
;  1253     1239  3             )
;  1254     1240  2         END;
;  1255     1241  2
;  1256     1242  2     [EVC$C_SRC_ARE]:                                ! Copy area number
;  1257     1243  3         BEGIN
;  1258     1244  3         CH$WCHAR_A (CH$RCHAR(RAW [RAW$T_SRCID]), PTR);  ! Copy the area number
;  1259     1245  2         END;
;  1260     1246  2     TES;
;  1261     1247  2
;  1262     1248  2 !
;  1263     1249  2 !       Copy the associated data with the event.
```

```
;  1264     1250  2 !
;  1265     1251  2
;  1266     1252  2     PTR = CH$MOVE(
;  1267     1253  2             .RAW [RAW$W_BYTES] - $BYTEOFFSET(RAW$T_DATA),
;  1268     1254  2             RAW [RAW$T_DATA],
;  1269     1255  2             .PTR);
;  1270     1256  2
;  1271     1257  2 !
;  1272     1258  2 !       Compute size of event in dna format, allocate a dbk for it
;  1273     1259  2 !       and copy it to the block.  The dbk is queued at the end of the
;  1274     1260  2 !       list in sink block for transmission.
;  1275     1261  2 !
;  1276     1262  2
;  1277     1263  2     SIZE = .PTR - EVTBFR;                          ! Size of event
;  1278     1264  2     EVL$ALLOCDBK                                   ! Allocate a buffer
;  1279     1265  2         (
;  1280     1266  2         .SIZE + EVQ$C_SIZE,                        ! size of buffer
;  1281     1267  2         .SNK [SNK$L_EVTBL],                        ! Queue at end of queue
;  1282     1268  2         EVQ                                        ! Return address here
;  1283     1269  2         );
;  1284     1270  2     EVQ [EVQ$W_EVTSIZE] = .SIZE;                   ! Set its size
;  1285     1271  2     CH$MOVE                                        ! Copy the event to buffer
;  1286     1272  2         (
;  1287     1273  2         .SIZE,
;  1288     1274  2         EVTBFR,
;  1289     1275  2         EVQ [EVQ$T_EVENT]
;  1290     1276  2         );
;  1291     1277  2     SNK [SNK$W_EVTCNT] =                           ! Count another event
;  1292     1278  2         .SNK [SNK$W_EVTCNT] + 1;
;  1293     1279  2
;  1294     1280  2     EVTDSC [0] = .EVQ [EVQ$W_EVTSIZE];             ! Build descriptor of event
;  1295     1281  2     EVTDSC [1] = EVQ [EVQ$T_EVENT];
;  1296     1282  2     EVL$PRINTLOG                                   ! Log event if requested
;  1297     1283  2         (
;  1298     1284  2         $BITPOSITION (ELG$V_QUEEVT),               ! Bit controlling logging
;  1299     1285  2         %ASCID 'Queued event',                     ! Header text
;  1300     1286  2         SNK [SNK$L_SNKLEN],                        ! Ncb is extra info
;  1301     1287  2         EVTDSC                                     ! Descriptor of event
;  1302     1288  2         );
;  1303     1289  2
;  1304     1290  2     EVL$WRITEVT (.SNK)                             ! Start the output if needed
;  1305     1291  2
;  1306     1292  1     END;


                                                                    .PSECT  $PLIT$,NOWRT,NOEXE,2

                00  74  73  6F  4C  20  73  74  6E  65  76  45  0005C P.AAK:   .ASCII  \Events Lost\<0>
                                                      010E000B  00068 P.AAJ:   .LONG   17694731
                                                      00000000' 0006C         .ADDRESS P.AAK
                74  6E  65  76  65  20  64  65  75  65  75  51  00070 P.AAM:   .ASCII  \Queued event\
                                                      010E000C  0007C P.AAL:   .LONG   17694732
                                                      00000000' 00080         .ADDRESS P.AAM
```

```
                                                        .PSECT  $CODE$,NOWRT,2

                                                00FC 00000 EVL$QUEUEVT:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7                  ; 1080
                                    5E      FEBC    CE  9E 00002            MOVAB   -324(SP), SP
                                    57        04    AC  D0 00007            MOVL    RAWEVT, RAW           ; 1135
                                    53        18    AE  9E 0000B            MOVAB   EVTBFR, PTR           ; 1136
                                    56        08    AC  D0 0000F            MOVL    SNKBLK, SNK           ; 1137
                            00C8    8F        2C    A6  B1 00013            CMPW    44(SNK), #200         ; 1139
                                                    49  1B 00019            BLEQU   2$
                                    50        2F    A6  9A 0001B            MOVZBL  47(SNK), R0           ; 1143
                                    50              50  D2 0001F            MCOML   R0, R0
                                      0C    AC              50  D3 00022    BITL    R0, SNKMASK           ; 1146
                                                    01  12 00026            BNEQ    1$
                                                    04  00028              RET
                                                      0C    AE  9F 00029 1$:  PUSHAB  EVQ                 ; 1150
                                                      28    A6  DD 0002C      PUSHL   40(SNK)               ; 1152
                                    51                50  D2 0002F        MCOML   R0, R1                ; 1151
                                7E    0C    AC              51  CB 00032    BICL3   R1, SNKMASK, -(SP)
                                    0000V   CF              03  FB 00037    CALLS   #3, EVL$BLDLOSTEVENT
                                      2F    A6        0C    AC  88 0003C    BISB2   SNKMASK, 47(SNK)      ; 1156
                                                      2C    A6  B6 00041    INCW    44(SNK)               ; 1158
                                    50                0C    AE  D0 00044    MOVL    EVQ, R0               ; 1159
                                      10    AE        0A    A0  3C 00048    MOVZWL  10(R0), EVTDSC
                                      14    AE        0C    A0  9E 0004D    MOVAB   12(R0), EVTDSC+4      ; 1160
                                                      10    AE  9F 00052    PUSHAB  EVTDSC                ; 1165
                                                      30    A6  9F 00055    PUSHAB  48(SNK)
                                                    0000'   CF  9F 00058    PUSHAB  P.AAJ                 ; 1163
                                                        05  DD 0005C       PUSHL   #5                    ; 1165
                                    0000G   CF              04  FB 0005E    CALLS   #4, EVL$PRINTLOG
                                                    04  00063              RET                           ; 1141
                                                      2F    A6  94 00064 2$:  CLRB    47(SNK)           ; 1173
                                    0122    8F              67  B1 00067    CMPW    (RAW), #290           ; 1178
                                                    05  1A 0006C            BGTRU   3$
                                            1E              67  B1 0006E    CMPW    (RAW), #30            ; 1180
                                                    0E  1E 00071            BGEQU   4$
                                        00000000G   8F      DD 00073 3$:    PUSHL   #EVL$_EVTSIZ          ; 1183
                                00000000G   00              01  FB 00079    CALLS   #1, LIB$SIGNAL
                                                    04  00080              RET                           ; 1182
                                                      08    AE  9F 00081 4$:  PUSHAB  MSECS               ; 1190
                                                      08    AE  9F 00084      PUSHAB  SECS
                                                      08    AE  9F 00087      PUSHAB  HALFDAY
                                                      02    A7  9F 0008A      PUSHAB  2(RAW)
                                    0000G   CF              04  FB 0008D    CALLS   #4, EVL$JULIAN
                                            83              01  90 00092    MOVB    #1, (PTR)+            ; 1198
                                            83        0C    AC  90 00095    MOVB    SNKMASK, (PTR)+       ; 1199
                            63        0A    A7              02  28 00099    MOVC3   #2, 10(RAW), (PTR)    ; 1207
                            63              6E              02  28 0009E    MOVC3   #2, HALFDAY, (R3)
                            63        04    AE              02  28 000A2    MOVC3   #2, SECS, (R3)
                            63        08    AE              02  28 000A7    MOVC3   #2, MSECS, (R3)
                                    50        0000'   CF      9A 000AC      MOVZBL  EVL$GB_LOCALNODE, R0  ; 1212
                            63    0000'   CF              50  28 000B1    MOVC3   R0, EVL$GT_LOCALNODE, (PTR)  ; 1214
                                    50        0C    A7      9A 000B7      MOVZBL  12(RAW), R0           ; 1217
                                            83              50  90 000BB    MOVB    R0, (PTR)+
                                      FF    8F              50  91 000BE    CMPB    R0, #255              ; 1221
                                                    32  13 000C2            BEQL    9$
                                            01              50  91 000C4    CMPB    R0, #1                ; 1222
```

```
                                 0A  13 000C7       BEQL    5$                          ;
                    03           50  91 000C9       CMPB    R0, #3                      ;
                                 0D  1F 000CC       BLSSU   6$                          ;
                    04           50  91 000CE       CMPB    R0, #4                      ;
                                 08  1A 000D1       BGTRU   6$                          ;
                    50      0D   A7  9A 000D3 5$:   MOVZBL  13(RAW), R0                 ; 1227
                                 50  D6 000D7       INCL    R0                          ;
                                 0B  11 000D9       BRB     7$                          ; 1229
                                 50  D5 000DB 6$:   TSTL    R0                          ; 1232
                                 0E  12 000DD       BNEQ    8$                          ;
                    50      0F   A7  9A 000DF       MOVZBL  15(RAW), R0                 ; 1236
                    50           03  C0 000E3       ADDL2   #3, R0                      ;
       63      0D   A7           50  28 000E6 7$:   MOVC3   R0, 13(RAW), (PTR)          ; 1238
                                 09  11 000EB       BRB     9$                          ; 1233
                    05           50  91 000ED 8$:   CMPB    R0, #5                      ; 1242
                                 04  12 000F0       BNEQ    9$                          ;
                    83      0D   A7  90 000F2       MOVB    13(RAW), (PTR)+             ; 1244
                    50           67  3C 000F6 9$:   MOVZWL  (RAW), R0                   ; 1253
                    50           1E  C2 000F9       SUBL2   #30, R0                     ;
       63      1E   A7           50  28 000FC       MOVC3   R0, 30(RAW), (PTR)          ; 1255
                    50      18   AE  9E 00101       MOVAB   EVTBFR, R0                  ; 1263
       52           53           50  C3 00105       SUBL3   R0, PTR, SIZE               ;
                            0C   AE  9F 00109       PUSHAB  EVQ                         ; 1265
                            28   A6  DD 0010C       PUSHL   40(SNK)                     ; 1267
                            0C   A2  9F 0010F       PUSHAB  12(SIZE)                    ; 1266
               0000V   CF        03  FB 00112       CALLS   #3, EVL$ALLOCDBK            ;
                    57      0C   AE  D0 00117       MOVL    EVQ, R7                     ; 1270
               0A   A7           52  B0 0011B       MOVW    SIZE, 10(R7)                ;
  0C   A7      18   AE           52  28 0011F       MOVC3   SIZE, EVTBFR, 12(R7)        ; 1275
                            2C   A6  B6 00125       INCW    44(SNK)                     ; 1278
               10   AE      0A   A7  3C 00128       MOVZWL  10(R7), EVTDSC              ; 1280
               14   AE      0C   A7  9E 0012D       MOVAB   12(R7), EVTDSC+4            ; 1281
                            10   AE  9F 00132       PUSHAB  EVTDSC                      ; 1286
                            30   A6  9F 00135       PUSHAB  48(SNK)                     ;
                         0000'   CF  9F 00138       PUSHAB  P.AAL                       ; 1284
                                 05  DD 0013C       PUSHL   #5                          ; 1286
               0000G   CF        04  FB 0013E       CALLS   #4, EVL$PRINTLOG            ;
                                 56  DD 00143       PUSHL   SNK                         ; 1290
               0000V   CF        01  FB 00145       CALLS   #1, EVL$WRITEEVT            ;
                                     04 0014A       RET                                 ; 1292

; Routine Size:  331 bytes,    Routine Base:  $CODE$ + 04F7
```

```
;  1308          1293  1 %,BTTL  'EVL$BLDLOSTEVENT  Build a Lost Event Event'
;  1309          1294  1 ROUTINE EVL$BLDLOSTEVENT (SNKMASK, PRED, EVQRTN) :NOVALUE =
;  1310          1295  1
;  1311          1296  1 !++
;  1312          1297  1 ! FUNCTIONAL DESCRIPTION:
;  1313          1298  1 !
;  1314          1299  1 !       Build a lost event event and queue it to a sink block.  The sink
;  1315          1300  1 !       mask for the lost event is passed here.  The current system time
;  1316          1301  1 !       is included in the event.
;  1317          1302  1 !
;  1318          1303  1 ! FORMAL PARAMETERS:
;  1319          1304  1 !
;  1320          1305  1 !       SNKMASK         Value of sink mask for event
;  1321          1306  1 !       PRED            Predecessor for insert into queue
;  1322          1307  1 !       EVQRTN          Address to return address of event block
;  1323          1308  1 !
;  1324          1309  1 ! IMPLICIT INPUTS:
;  1325          1310  1 !
;  1326          1311  1 !       NONE
;  1327          1312  1 !
;  1328          1313  1 ! IMPLICIT OUTPUTS:
;  1329          1314  1 !
;  1330          1315  1 !       NONE
;  1331          1316  1 !
;  1332          1317  1 ! ROUTINE VALUE:
;  1333          1318  1 ! COMPLETION CODES:
;  1334          1319  1 !
;  1335          1320  1 !       NONE
;  1336          1321  1 !
;  1337          1322  1 ! SIDE EFFECTS:
;  1338          1323  1 !
;  1339          1324  1 !       NONE
;  1340          1325  1 !
;  1341          1326  1 !--
;  1342          1327  1
;  1343          1328  2     BEGIN
;  1344          1329  2
;  1345          1330  2     LOCAL
;  1346          1331  2         EVQ     : REF BBLOCK,           ! Pointer to event block
;  1347          1332  2         PTR,                            ! Pointer into event block
;  1348          1333  2         SYSTIM  : VECTOR [2],           ! 64 bit system time (now)
;  1349          1334  2         HALFDAY,                        ! Julian half day for now
;  1350          1335  2         SECS,                           ! and the rest
;  1351          1336  2         MSECS                           ! of the dna time
;  1352          1337  2         ;
;  1353          1338  2
;  1354          1339  2     EVL$ALLOCDBK                        ! Obtain an event block
;  1355          1340  2         (
;  1356          1341  2         EVQ$C_SIZE + 22,                ! Thats big enough
;  1357          1342  2         .PRED,                          ! In correct queue
;  1358          1343  2         EVQ                             ! Return address here
;  1359          1344  2         );
;  1360          1345  2
;  1361        P 1346  2     $GETTIM                             ! When is now
;  1362        P 1347  2         (
;  1363        P 1348  2         TIMADR = SYSTIM
;  1364          1349  2         );
```

```
;  1365          1350  2
;  1366          1351  2      EVL$JULIAN                                      ! When is it really according
;  1367          1352  2          (                                           ! to DECnet.
;  1368          1353  2          SYSTIM,                                     ! 64 bit time
;  1369          1354  2          HALFDAY, SECS, MSECS                        ! Dna time
;  1370          1355  2          );
;  1371          1356  2
;  1372          1357  2      PTR = EVQ [EVQ$T_EVENT];                        ! Point to place to build evt
;  1373          1358  2
;  1374          1359  2      CH$WCHAR_A (1, PTR);                            ! Just a code byte for event
;  1375          1360  2      CH$WCHAR_A (.$NKMASK, PTR);                     ! The sink mask byte
;  1376          1361  2      PTR = CH$COPY                                   ! Build more of the header
;  1377          1362  2          (
;  1378          1363  2          2, UPLIT WORD (EVC$C_NMA_LOS),              ! Event code
;  1379          1364  2          2, HALFDAY,                                 ! Date and time in dna format
;  1380          1365  2          2, SECS,
;  1381          1366  2          2, MSECS,
;  1382          1367  2          0,                                          ! Just a dummy fill char
;  1383          1368  2          8, .PTR                                     ! Place to put it
;  1384          1369  2          );
;  1385          1370  2
;  1386          1371  2      PTR = CH$MOVE                                   ! Put in the source node
;  1387          1372  2          (                                           ! in node id format
;  1388          1373  2          .EVL$GB_LOCALNODE,                          ! from a special buffer
;  1389          1374  2          EVL$GT_LOCALNODE,                           ! Built in advance
;  1390          1375  2          .PTR
;  1391          1376  2          );
;  1392          1377  2
;  1393          1378  2      CH$WCHAR_A (EVC$C_SRC_NON, PTR);                ! Specify the source type
;  1394          1379  2
;  1395          1380  2      EVQ [EVQ$W_EVTSIZE] = .PTR - EVQ [EVQ$T_EVENT]; ! Set the size
;  1396          1381  2      .EVQRTN = .EVQ                                  ! Return the address
;  1397          1382  2
;  1398          1383  1      END;


                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                                0000  00084 P.AAN:  .WORD   0                                                    ;

                                                .EXTRN  SYS$GETTIM

                                                .PSECT  $CODE$,NOWRT,2

                                00FC  00000 EVL$BLDLOSTEVENT:
                                                .WORD   Save R2,R3,R4,R5,R6,R7                                  ; 1294
                          5E            18  C2 00002          SUBL2   #24, SP                                   ;
                                        5E  DD 00005          PUSHL   SP                                        ; 1340
                                    08  AC  DD 00007          PUSHL   PRED                                      ; 1342
                                        22  DD 0000A          PUSHL   #34                                       ; 1341
                      0000V     CF          03  FB 0000C      CALLS   #3, EVL$ALLOCDBK                          ;
                                    10  AE  9F 00011          PUSHAB  SYSTIM                                    ; 1349
                  00000000G     00          01  FB 00014      CALLS   #1, SYS$GETTIM                            ;
                                    0C  AE  9F 0001B          PUSHAB  MSECS                                     ; 1352
                                    0C  AE  9F 0001E          PUSHAB  SECS                                      ;
                                    0C  AE  9F 00021          PUSHAB  HALFDAY                                   ;
```

```
                                        1C    AE  9F 00024         PUSHAB  SYSTIM                                          ;
                              0000G     CF          04  FB 00027   CALLS   #4, EVL$JULIAN                                ;
                                    56            6E  D0 0002C   MOVL    EVQ, R6                                       ; 1357
                                    57    0C      A6  9E 0002F   MOVAB   12(R6), R7                                      ;
                                    53            57  D0 00033   MOVL    R7, PTR                                       ;
                                    83            01  90 00036   MOVB    #1, (PTR)+                                    ; 1359
                                    83    04      AC  90 00039   MOVB    SNKMASK, (PTR)+                               ; 1360
                    63      0000'   CF          02  28 0003D   MOVC3   #2, P.AAN, (PTR)                              ; 1368
                    63        04    AE          02  28 00043   MOVC3   #2, HALFDAY, (R3)                              ;
                    63        08    AE          02  28 00048   MOVC3   #2, SECS, (R3)                                 ;
                    63        0C    AE          02  28 0004D   MOVC3   #2, MSECS, (R3)                                ;
                                    50  0000'   CF  9A 00052   MOVZBL  EVL$GB_LOCALNODE, R0                          ; 1373
                    63      0000'   CF          50  28 00057   MOVC3   R0, EVL$GT_LOCALNODE, (PTR)                    ; 1375
                                    83          01  8E 0005D   MNEGB   #1, (PTR)+                                    ; 1378
                0A    A6            53          57  A3 00060   SUBW3   R7, PTR, 10(R6)                               ; 1380
                              0C    BC          56  D0 00065   MOVL    R6, @EVQRTN                                   ; 1381
                                                    04 00069   RET                                                   ; 1383

; Routine Size:  106 bytes,    Routine Base:  $CODE$ + 0642
```

```
;  1400     1384  1 %SBTTL  'EVL$NETERROR  Check and Report a Network Error'
;  1401     1385  1 ROUTINE EVL$NETERROR (CODE, STATUSVAL, IOSBADR) =
;  1402     1386  1
;  1403     1387  1 !++
;  1404     1388  1 ! FUNCTIONAL DESCRIPTION:
;  1405     1389  1 !
;  1406     1390  1 !       Check the status value and/or the iosb status code of
;  1407     1391  1 !       an io request to the network.  If anything is amiss then
;  1408     1392  1 !       signal the error to the log and return failure.  If all is
;  1409     1393  1 !       well, return success.
;  1410     1394  1 !
;  1411     1395  1 ! FORMAL PARAMETERS:
;  1412     1396  1 !
;  1413     1397  1 !       CODE            Value of the EVL condition code to signal
;  1414     1398  1 !       STATUSVAL       Value of status from a network request
;  1415     1399  1 !       IOSBADR         Address of the iosb of a network request
;  1416     1400  1 !                       a value of zero implies do not check the iosb
;  1417     1401  1 !
;  1418     1402  1 ! IMPLICIT INPUTS:
;  1419     1403  1 !
;  1420     1404  1 !       NONE
;  1421     1405  1 !
;  1422     1406  1 ! IMPLICIT OUTPUTS:
;  1423     1407  1 !
;  1424     1408  1 !       NONE
;  1425     1409  1 !
;  1426     1410  1 ! ROUTINE VALUE:
;  1427     1411  1 ! COMPLETION CODES:
;  1428     1412  1 !
;  1429     1413  1 !       Success if no error, failure otherwise.
;  1430     1414  1 !       Any errors are signaled so they will be logged.
;  1431     1415  1 !
;  1432     1416  1 ! SIDE EFFECTS:
;  1433     1417  1 !
;  1434     1418  1 !       NONE
;  1435     1419  1 !
;  1436     1420  1 !--
;  1437     1421  1
;  1438     1422  2     BEGIN
;  1439     1423  2
;  1440     1424  2     MAP
;  1441     1425  2         IOSBADR : REF BBLOCK                    ! Its the address of a block
;  1442     1426  2         ;
;  1443     1427  2
;  1444     1428  2     LOCAL
;  1445     1429  2         IOSBSTS                                 ! Hold its value here
;  1446     1430  2         ;
;  1447     1431  2     IF NOT .STATUSVAL                           ! Signal any error here
;  1448     1432  2     THEN
;  1449     1433  2         SIGNAL (.CODE, 0, .STATUSVAL)
;  1450     1434  2     ;
;  1451     1435  2
;  1452     1436  2     IOSBSTS =                                   ! Obtain the value
;  1453     1437  3         BEGIN
;  1454     1438  3         IF .IOSBADR EQL 0                       ! Unless there is no block
;  1455     1439  3         THEN
;  1456     1440  3             SUCCESS                             ! in which case use success
```

```
;  1457          1441  3          ELSE
;  1458          1442  3              .IOSBADR [0, 0, 16, 0]              ! Status value of iosb
;  1459          1443  3          END;
;  1460          1444  2
;  1461          1445  2      IF NOT .IOSBSTS                             ! If its bad, signal it
;  1462          1446  2      THEN
;  1463          1447  2          SIGNAL (.CODE, 0, .IOSBSTS)
;  1464          1448  2      ;
;  1465          1449  2
;  1466          1450  2      RETURN .STATUSVAL AND .IOSBSTS AND SUCCESS  ! and return problems
;  1467          1451  2
;  1468          1452  1      END;


                                        000C 00000 EVL$NETERROR:
                                                            .WORD   Save R2,R3                                      ; 1385
                                53 00000000G   00    9E 00002       MOVAB   LIB$SIGNAL, R3                          ;
                                0B         08  AC    E8 00009       BLBS    STATUSVAL, 1$                           ; 1431
                                           08  AC    DD 0000D       PUSHL   STATUSVAL                               ; 1433
                                               7E    D4 00010       CLRL    -(SP)                                   ;
                                           04  AC    DD 00012       PUSHL   CODE                                    ;
                                63             03    FB 00015       CALLS   #3, LIB$SIGNAL                          ;
                                           0C  AC    D5 00018 1$:   TSTL    IOSBADR                                 ; 1438
                                               05    12 0001B       BNEQ    2$                                      ;
                                52             01    D0 0001D       MOVL    #1, IOSBSTS                             ;
                                               04    11 00020       BRB     3$                                      ;
                                52         0C  BC    3C 00022 2$:   MOVZWL  @IOSBADR, IOSBSTS                       ; 1442
                                0A             52    E8 00026 3$:   BLBS    IOSBSTS, 4$                             ; 1445
                                               52    DD 00029       PUSHL   IOSBSTS                                 ; 1447
                                               7E    D4 0002B       CLRL    -(SP)                                   ;
                                           04  AC    DD 0002D       PUSHL   CODE                                    ;
                                   63          03    FB 00030       CALLS   #3, LIB$SIGNAL                          ;
                                   50          52    D2 00033 4$:   MCOML   IOSBSTS, R0                             ; 1450
                          50           08  AC        50    CB 00036       BICL3   R0, STATUSVAL, R0                   ;
                50        50              01        00    EF 0003B       EXTZV   #0, #1, R0, R0                      ;
                                                      04    00040       RET                                         ; 1452

; Routine Size:  65 bytes,    Routine Base:  $CODE$ + 06AC
```

```
;  1470     1453  1 %SBTTL  'EVL$WRITEVT  Write an Event to a Sink Node'
;  1471     1454  1 ROUTINE EVL$WRITEVT (SNKBLK) :NOVALUE =
;  1472     1455  1
;  1473     1456  1 !++
;  1474     1457  1 ! FUNCTIONAL DESCRIPTION:
;  1475     1458  1 !
;  1476     1459  1 !       If a write is in progress, just return from here.
;  1477     1460  1 !       If the event queue is empty, close and delete the sink
;  1478     1461  1 !       if the status bits are set, but in any case return from here.
;  1479     1462  1 !       If the sink is not open, open it.  We will return here when
;  1480     1463  1 !       the open completes.
;  1481     1464  1 !       If everything is go, then start the write on the open logical
;  1482     1465  1 !       link to the event receiver and check for any errors on the qio.
;  1483     1466  1 !       The remainder of the write is handled in the WRITEDONE routine
;  1484     1467  1 !       which is activated as work when the AST completes.
;  1485     1468  1 !
;  1486     1469  1 !       When we find the event queue empty, we set a timer which
;  1487     1470  1 !       will activate the close routine.  The link will be reopened
;  1488     1471  1 !       when an event is requeued.  If an event is written we cancel the
;  1489     1472  1 !       timer entry.
;  1490     1473  1 !
;  1491     1474  1 ! FORMAL PARAMETERS:
;  1492     1475  1 !
;  1493     1476  1 !       SNKBLK          Address of the sink block
;  1494     1477  1 !
;  1495     1478  1 ! IMPLICIT INPUTS:
;  1496     1479  1 !
;  1497     1480  1 !       NONE
;  1498     1481  1 !
;  1499     1482  1 ! IMPLICIT OUTPUTS:
;  1500     1483  1 !
;  1501     1484  1 !       NONE
;  1502     1485  1 !
;  1503     1486  1 ! ROUTINE VALUE:
;  1504     1487  1 ! COMPLETION CODES:
;  1505     1488  1 !
;  1506     1489  1 !       NONE
;  1507     1490  1 !
;  1508     1491  1 ! SIDE EFFECTS:
;  1509     1492  1 !
;  1510     1493  1 !       NONE
;  1511     1494  1 !
;  1512     1495  1 !--
;  1513     1496  1
;  1514     1497  2     BEGIN
;  1515     1498  2
;  1516     1499  2     LOCAL
;  1517     1500  2         SNK     : REF BBLOCK,                   ! Pointer to sink block
;  1518     1501  2         EVQ     : REF BBLOCK,                   ! Pointer to event queue block
;  1519     1502  2         STATUS                                  ! Local status value
;  1520     1503  2         ;
;  1521     1504  2
;  1522     1505  2     SNK = .SNKBLK;                              ! Point to sink block
;  1523     1506  2
;  1524     1507  2     IF .SNK [SNK$V_STS_BSY]                     ! If some action in progress
;  1525     1508  2     THEN
;  1526     1509  2         RETURN                                  ! Go away and wait for done
```

```
;  1527     1510  2      ;
;  1528     1511  2
;  1529     1512  2      IF  .SNK [SNK$L_EVTFL]                      ! If event list is empty
;  1530     1513  2          EQL
;  1531     1514  2          SNK [SNK$L_EVTFL]
;  1532     1515  2      THEN
;  1533     1516  3          BEGIN
;  1534     1517  3          IF  .SNK [SNK$V_STS_DEL]                ! If delete or close set
;  1535     1518  3              OR
;  1536     1519  3              .SNK [SNK$V_STS_CLS]
;  1537     1520  3          THEN
;  1538     1521  3              EVL$CLOSESNK (.SNK)                 ! Close and deallocate sink
;  1539     1522  3          ELSE
;  1540     1523  4              BEGIN
;  1541     1524  4              SNK [SNK$V_STS_TMR] = TRUE;         ! Set bit for timer active
;  1542     1525  4              WKQ$ADD_TIMED_WORK                  ! Set work to perform close
;  1543     1526  4                  (
;  1544     1527  4                  EVL$CLOSESNK,                   ! Close after the delay
;  1545     1528  4                  .SNK, 0,                        ! Sink is the parameter
;  1546     1529  4                  EVL$Q_CLOSEDELAY,               ! Time delay
;  1547     1530  4                  .SNK                            ! Request id is sink address
;  1548     1531  4                  )
;  1549     1532  4              END
;  1550     1533  3          ;
;  1551     1534  3          RETURN                                  ! and we are done here
;  1552     1535  3          END
;  1553     1536  2      ;
;  1554     1537  2
;  1555     1538  2      IF .SNK [SNK$V_STS_TMR]                     ! If close timer is active
;  1556     1539  2      THEN
;  1557     1540  3          BEGIN
;  1558     1541  3          WKQ$CANCEL_TIMED_WORK (.SNK);           ! Cancel the work element
;  1559     1542  3          SNK [SNK$V_STS_TMR] = FALSE             ! Clear the bit
;  1560     1543  3          END
;  1561     1544  2      ;
;  1562     1545  2
;  1563     1546  2      EVQ = .SNK [SNK$L_EVTFL];                   ! Point to the first event
;  1564     1547  2      SNK [SNK$L_ROUTINE] = EVL$WRITEDONE;        ! Setup the routine to go to
;  1565     1548  2
;  1566     1549  2      IF .SNK [SNK$L_SNKADR] EQL .EVL$GT_LOCALNODE<0,16>  ! If to local receiver,
;  1567     1550  2      THEN
;  1568     1551  3          BEGIN
;  1569     1552  3          SNK [SNK$V_STS_BSY] = TRUE;             ! Write is in progress
;  1570     1553  3          WKQ$ADD_WORK_ITEM(EVL$QUEUE_EVENT,      ! Queue event to local receiver
;  1571     1554  3              .EVQ [EVQ$W_EVTSIZE], EVQ [EVQ$T_EVENT]);
;  1572     1555  3          SNK [SNK$W_IOSB] = TRUE;                ! Set write successful
;  1573     1556  3          EVL$ASTWORK(.SNK);                      ! and simulate I/O completion
;  1574     1557  3          RETURN;
;  1575     1558  2          END;
;  1576     1559  2
;  1577     1560  2      IF NOT .SNK [SNK$V_STS_OPN]                 ! If sink is not open
;  1578     1561  2      THEN
;  1579     1562  3          BEGIN
;  1580     1563  3          EVL$OPENSNK (.SNK);                     ! Open the sink node link
;  1581     1564  3          RETURN                                  ! we will come back here later
;  1582     1565  2          END;
;  1583     1566  2
```

```
P 1567  2       STATUS = $QIO                                     ! Do the write
P 1568  2           (
P 1569  2           EFN = EVL$C_ASYNCH_EFN,
P 1570  2           CHAN = .SNK [SNK$W_NETCHAN],                  ! Channel in sink block
P 1571  2           FUNC = IO$_WRITEVBLK,                         ! Write data to link
P 1572  2           IOSB = SNK [SNK$W_IOSB],                      ! Leave status here
P 1573  2           ASTADR = EVL$ASTWORK,                         ! Our general ast routine
P 1574  2           ASTPRM = .SNK,                                ! Sink block is parameter
P 1575  2           P1 = EVQ [EVQ$T_EVENT],                       ! Address of event
P 1576  2           P2 = .EVQ [EVQ$W_EVTSIZE]                     ! Size of event in bytes
  1577  2           );
  1578  2
  1579  2       IF NOT EVL$NETERROR (EVL$_WRTEVT, .STATUS, 0) ! Report any error
  1580  2       THEN
  1581  2           EVL$CLOSESNK (.SNK)                           ! Close the sink on an error
  1582  2       ELSE
  1583  2           SNK [SNK$V_STS_BSY] = TRUE                    ! Write is in progress
  1584  2
  1585  1       END;



                                              001C 00000 EVL$WRITEVT:
                                                                .WORD   Save R2,R3,R4                              ; 1454
                        52      04  AC  D0 00002                MOVL    SNKBLK, SNK                                ; 1505
                        54      2E  A2  9E 00006                MOVAB   46(SNK), R4                                ; 1507
              01        64          01  E1 0000A                BBC     #1, (R4), 1$                               ;
                                        04 0000E                RET                                                ;
                        50      24  A2  9E 0000F 1$:            MOVAB   36(SNK), R0                                ; 1514
                        50      24  A2  D1 00013                CMPL    36(SNK), R0                                ;
                                    22  12 00017                BNEQ    4$                                         ;
              03        64          03  E1 00019                BBC     #3, (R4), 3$                               ; 1517
                                  009C  31 0001D 2$:            BRW     8$                                         ;
              F9        64          04  E0 00020 3$:            BBS     #4, (R4), 2$                               ; 1519
                        64          20  88 00024                BISB2   #32, (R4)                                  ; 1524
                                    52  DD 00027                PUSHL   SNK                                        ; 1530
                             0000'  CF  9F 00029                PUSHAB  EVL$Q_CLOSEDELAY                           ; 1526
                                    7E  D4 0002D                CLRL    -(SP)                                      ;
                                    52  DD 0002F                PUSHL   SNK                                        ; 1528
                             0000V  CF  9F 00031                PUSHAB  EVL$CLOSESNK                               ; 1526
                  0000G  CF         05  FB 00035                CALLS   #5, WKQ$ADD_TIMED_WORK                     ;
                                    04     0003A                RET                                                ; 1516
              0A        64          05  E1 0003B 4$:            BBC     #5, (R4), 5$                               ; 1538
                                    52  DD 0003F                PUSHL   SNK                                        ; 1541
                  0000G  CF         01  FB 00041                CALLS   #1, WKQ$CANCEL_TIMED_WORK                  ;
                        64          20  8A 00046                BICB2   #32, (R4)                                  ; 1542
                        53      24  A2  D0 00049 5$:            MOVL    36(SNK), EVQ                               ; 1546
                  14    A2   0000V  CF  9E 0004D                MOVAB   EVL$WRITEDONE, 20(SNK)                     ; 1547
   18   A2   0000'  CF  10          00  ED 00053                CMPZV   #0, #16, EVL$GT_LOCALNODE, 24(SNK)         ; 1549
                                    1F  12 0005B                BNEQ    6$                                         ;
                        64          02  88 0005D                BISB2   #2, (R4)                                   ; 1552
                                0C  A3  9F 00060                PUSHAB  12(EVQ)                                    ; 1554
                        7E      0A  A3  3C 00063                MOVZWL  10(EVQ), -(SP)                             ;
                             0000G  CF  9F 00067                PUSHAB  EVL$QUEUE_EVENT                            ; 1553
                  0000G  CF         03  FB 0006B                CALLS   #3, WKQ$ADD_WORK_ITEM                      ; 1554
```

```
                          0C    A2              01  B0 00070          MOVW    #1, 12(SNK)                                 ; 1555
                                                52  DD 00074          PUSHL   SNK                                         ; 1556
                        0000V   CF              01  FB 00076          CALLS   #1, EVL$ASTWORK
                                                04  0007B          RET                                                 ; 1551
                                08              64  E8 0007C 6$:      BLBS    (R4), 7$                                    ; 1560
                                                52  DD 0007F          PUSHL   SNK                                         ; 1563
                        0000V   CF              01  FB 00081          CALLS   #1, EVL$OPENSNK
                                                04  00086          RET                                                 ; 1562
                                                7E  7C 00087 7$:      CLRQ    -(SP)                                       ; 1577
                                                7E  7C 00089          CLRQ    -(SP)
                                7E      0A      A3  3C 0008B          MOVZWL  10(EVQ), -(SP)
                                        0C      A3  9F 0008F          PUSHAB  12(EVQ)
                                                52  DD 00092          PUSHL   SNK
                                    0000V       CF  9F 00094          PUSHAB  EVL$ASTWORK
                                        0C      A2  9F 00098          PUSHAB  12(SNK)
                                                30  DD 0009B          PUSHL   #48
                                7E      0A      A2  3C 0009D          MOVZWL  10(SNK), -(SP)
                                                02  DD 000A1          PUSHL   #2
                    000C0000G   00              0C  FB 000A3          CALLS   #12, SYS$QIO
                                                7E  D4 000AA          CLRL    -(SP)                                       ; 1579
                                                50  DD 000AC          PUSHL   STATUS
                                00000000G       8F  DD 000AE          PUSHL   #EVL$_WRTEVT
                        FF06    CF              03  FB 000B4          CALLS   #3, EVL$NETERROR
                                08              50  E8 000B9          BLBS    R0, 9$
                                                52  DD 000BC 8$:      PUSHL   SNK                                         ; 1581
                        0000V   CF              01  FB 000BE          CALLS   #1, EVL$CLOSESNK
                                                04  000C3          RET
                                64              02  88 000C4 9$:      BISB2   #2, (R4)                                    ; 1583
                                                04  000C7          RET                                                 ; 1585

; Routine Size:  200 bytes,    Routine Base:  $CODE$ + 06ED
```

```
;  1604          1586  1 %SBTTL  'EVL$WRITEDONE  Finish Up an Event Write'
;  1605          1587  1 ROUTINE EVL$WRITEDONE (SNKBLK) :NOVALUE =
;  1606          1588  1
;  1607          1589  1 !++
;  1608          1590  1 ! FUNCTIONAL DESCRIPTION:
;  1609          1591  1 !
;  1610          1592  1 !       This routine is queued as work by the ast completion of
;  1611          1593  1 !       a write of an event to a sink.  The parameter is the sink block.
;  1612          1594  1 !       Check for an error on the write and if there was none, then
;  1613          1595  1 !       dequeue the event.  If there was an error, close the sink.
;  1614          1596  1 !       This will cause the sink to be reopened, creating a new logical
;  1615          1597  1 !       link and a retransmission will occur.
;  1616          1598  1 !
;  1617          1599  1 ! FORMAL PARAMETERS:
;  1618          1600  1 !
;  1619          1601  1 !       SNKBLK          Address of the sink block of concern
;  1620          1602  1 !
;  1621          1603  1 ! IMPLICIT INPUTS:
;  1622          1604  1 !
;  1623          1605  1 !       NONE
;  1624          1606  1 !
;  1625          1607  1 ! IMPLICIT OUTPUTS:
;  1626          1608  1 !
;  1627          1609  1 !       NONE
;  1628          1610  1 !
;  1629          1611  1 ! ROUTINE VALUE:
;  1630          1612  1 ! COMPLETION CODES:
;  1631          1613  1 !
;  1632          1614  1 !       NONE
;  1633          1615  1 !
;  1634          1616  1 ! SIDE EFFECTS:
;  1635          1617  1 !
;  1636          1618  1 !       NONE
;  1637          1619  1 !
;  1638          1620  1 !--
;  1639          1621  1
;  1640          1622  2     BEGIN
;  1641          1623  2
;  1642          1624  2     LOCAL
;  1643          1625  2         SNK     : REF BBLOCK                    ! Pointer to the sink block
;  1644          1626  2         ;
;  1645          1627  2
;  1646          1628  2     SNK = .SNKBLK;                              ! Set the pointer
;  1647          1629  2     SNK [SNK$V_STS_BSY] = FALSE;                ! Write is not in progress
;  1648          1630  2
;  1649          1631  2     IF EVL$NETERROR                             ! Check for an error
;  1650          1632  2         (EVL$_WRTEVT, TRUE, SNK [SNK$W_IOSB] )
;  1651          1633  2     THEN
;  1652          1634  3         BEGIN
;  1653          1635  3         EVL$DEALLOCDBK (.SNK [SNK$L_EVTFL] );   ! No error, dump that event
;  1654          1636  3         SNK [SNK$W_EVTCNT] =                    ! Reduce count of queued evts
;  1655          1637  3             .SNK [SNK$W_EVTCNT] - 1;
;  1656          1638  3         EVL$WRITEVT (.SNK)                      ! and write the next one
;  1657          1639  3         END
;  1658          1640  2     ELSE
;  1659          1641  3         BEGIN
;  1660          1642  3         EVL$CLOSESNK (.SNK)                     ! Close the sink to try again
```

```
; 1661          1643 3          END
; 1662          1644 3
; 1663          1645 1      END;



                                      0004 00000 EVL$WRITEDONE:
                                                        .WORD   Save R2                              ; 1587
                  52        04   AC  D0 00002           MOVL    SNKBLK, SNK                          ; 1628
         2E       A2             02  8A 00006           BICB2   #2, 46(SNK)                          ; 1629
                            0C   A2  9F 0000A           PUSHAB  12(SNK)                              ; 1632
                                 01  DD 0000D           PUSHL   #1                                   ;
                        00000000G 8F DD 0000F           PUSHL   #EVL$_WRTEVT                         ;
       FEDD       CF             03  FB 00015           CALLS   #3, EVL$NETERROR                     ;
                  13             50  E9 0001A           BLBC    R0, 1$                               ;
                            24   A2  DD 0001D           PUSHL   36(SNK)                              ; 1635
       0000V      CF             01  FB 00020           CALLS   #1, EVL$DEALLOCDBK                   ;
                            2C   A2  B7 00025           DECW    44(SNK)                              ; 1637
                                 52  DD 00028           PUSHL   SNK                                  ; 1638
       FF09       CF             01  FB 0002A           CALLS   #1, EVL$WRITEVT                      ;
                                     04 0002F           RET                                          ;
                                 52  DD 00030 1$:       PUSHL   SNK                                  ; 1642
       0000V      CF             01  FB 00032           CALLS   #1, EVL$CLOSESNK                     ;
                                     04 00037           RET                                          ; 1645

; Routine Size:  56 bytes,    Routine Base:  $CODE$ + 07B5
```

```
;  1665          1646  1 %SBTTL  'EVL$OPENSNK  Open a Link to a Sink Node'
;  1666          1647  1 ROUTINE EVL$OPENSNK (SNKBLK) :NOVALUE =
;  1667          1648  1
;  1668          1649  1 !++
;  1669          1650  1 ! FUNCTIONAL DESCRIPTION:
;  1670          1651  1 !
;  1671          1652  1 !       Open a logical link to the event receiver on the sink node.
;  1672          1653  1 !       If an error occurs here, close and delete the sink.  Something
;  1673          1654  1 !       is drastically wrong if there is an error here and we do not retry.
;  1674          1655  1 !       The access function for the link finishes up in EVL$OPENDONE
;  1675          1656  1 !       and further errors are handled there.
;  1676          1657  1 !
;  1677          1658  1 ! FORMAL PARAMETERS:
;  1678          1659  1 !
;  1679          1660  1 !       SNKBLK          Address of the sink block
;  1680          1661  1 !
;  1681          1662  1 ! IMPLICIT INPUTS:
;  1682          1663  1 !
;  1683          1664  1 !       NONE
;  1684          1665  1 !
;  1685          1666  1 ! IMPLICIT OUTPUTS:
;  1686          1667  1 !
;  1687          1668  1 !       NONE
;  1688          1669  1 !
;  1689          1670  1 ! ROUTINE VALUE:
;  1690          1671  1 ! COMPLETION CODES:
;  1691          1672  1 !
;  1692          1673  1 !       NONE
;  1693          1674  1 !
;  1694          1675  1 ! SIDE EFFECTS:
;  1695          1676  1 !
;  1696          1677  1 !       NONE
;  1697          1678  1 !
;  1698          1679  1 !--
;  1699          1680  1
;  1700          1681  2     BEGIN
;  1701          1682  2
;  1702          1683  2     LOCAL
;  1703          1684  2         SNK     : REF BBLOCK,                   ! Pointer to a sink block
;  1704          1685  2         STATUS                                  ! Status return
;  1705          1686  2         ;
;  1706          1687  2
;  1707          1688  2     SNK = .SNKBLK;                              ! Set local pointer
;  1708          1689  2     IF .SNK [SNK$V_STS_OPN]                     ! If its already open,
;  1709          1690  2         OR
;  1710          1691  2         .SNK [SNK$V_STS_BSY]                    ! or somethings going on
;  1711          1692  2     THEN
;  1712          1693  2         RETURN                                  ! Just ignore this call
;  1713          1694  2     ;
;  1714          1695  2
;  1715          1696  2     IF .EVL$GL_LOGMASK [ELG$V_SNKOPN]           ! Log open events?
;  1716          1697  2     THEN
;  1717          1698  3         BEGIN
;  1718          1699  3         SIGNAL                                  ! Log this open attempt
;  1719          1700  3             (
;  1720          1701  3             EVL$_LOGOPNT, 2,                    ! Message and 2 parms
;  1721          1702  3             SNK [SNK$L_SNKLEN],                 ! Ncb address
```

```
;  1722          1703  3              0                                   ! and zero for current time
;  1723          1704  3              )
;  1724          1705  3          END
;  1725          1706  2      ;
;  1726          1707  2
;  1727        P 1708  2      STATUS = $ASSIGN                            ! Assign a channel to net
;  1728        P 1709  2          (
;  1729        P 1710  2          DEVNAM = %ASCID '_NET:',
;  1730        P 1711  2          CHAN = SNK [SNK$W_NETCHAN]               ! Place channel in sink block
;  1731          1712  2          );
;  1732          1713  2
;  1733          1714  2      IF .STATUS                                  ! If that worked
;  1734          1715  2      THEN
;  1735          1716  3          BEGIN
;  1736          1717  3          SNK [SNK$V_STS_OPN] = TRUE;             ! We have a channel tied up
;  1737          1718  3          SNK [SNK$L_ROUTINE] = EVL$OPENDONE;     ! Set the work routine
;  1738        P 1719  3          STATUS = $QIO                           ! and open the logical link
;  1739        P 1720  3              (
;  1740        P 1721  3              EFN = EVL$C_ASYNCH_EFN,
;  1741        P 1722  3              CHAN = .SNK [SNK$W_NETCHAN],        ! The channel
;  1742        P 1723  3              FUNC = IO$_ACCESS,                  ! access a link
;  1743        P 1724  3              IOSB = SNK [SNK$W_IOSB],            ! Place status here
;  1744        P 1725  3              ASTADR = EVL$ASTWORK,               ! Use the command ast routine
;  1745        P 1726  3              ASTPRM = .SNK,                      ! and the sink block is param
;  1746        P 1727  3              P2 = SNK [SNK$L_SNKLEN]             ! The ncb is in the sink block
;  1747          1728  4              )
;  1748          1729  3          END
;  1749          1730  2      ;
;  1750          1731  2
;  1751          1732  2      IF NOT EVL$NETERROR (EVL$_OPNSNK, .STATUS, 0) ! If any error
;  1752          1733  2      THEN
;  1753          1734  3          BEGIN
;  1754          1735  3          SNK [SNK$V_STS_CLS] = TRUE;             ! Mark for delete and
;  1755          1736  3          EVL$CLOSESNK (.SNK)                     ! Close the link
;  1756          1737  3          END
;  1757          1738  2      ELSE
;  1758          1739  2          SNK [SNK$V_STS_BSY] = TRUE;             ! Mark as busy
;  1759          1740  2
;  1760          1741  1      END;


                                                    .PSECT  $PLIT$,NOWRT,NOEXE,2

                                            00086           .BLKB   2
                00  00  00  3A  54  45  4E  5F  00088 P.AAP:  .ASCII  \_NET:\<0><0><0>                    ;
                                  010E0005  00090 P.AAO:  .LONG   17694725                                ;
                                  00000000' 00094         .ADDRESS P.AAP                                  ;

                                                    .EXTRN  SYS$ASSIGN

                                                    .PSECT  $CODE$,NOWRT,2

                                      000C 00000 EVL$OPENSNK:
                                                    .WORD   Save R2,R3                                      ; 1647
                          52        04   AC  D0 00002         MOVL    SNKBLK, SNK                         ; 1688
                          53        2E   A2  9E 00006         MOVAB   46(SNK), R3                         ; 1689
```

```
                                 7B              63  E8 0000A          BLBS    (R3), 4$                                        ;
                          77     63              01  E0 0000D          BBS     #1, (R3), 4$                                    ; 1691
                          14     0000G   CF      01  E1 00011          BBC     #1, EVL$GL_LOGMASK, 1$                          ; 1696
                                                 7E  D4 00017          CLRL    -(SP)                                           ; 1702
                                         30      A2  9F 00019          PUSHAB  48(SNK)                                         ;
                                                 02  DD 0001C          PUSHL   #2                                              ; 1700
                                 00000000G       8F  DD 0001E          PUSHL   #EVL$_LOGOPNT                                   ;
                             00000000G   00      04  FB 00024          CALLS   #4, LIB$SIGNAL                                  ;
                                                 7E  7C 0002B 1$:      CLRQ    -(SP)                                           ; 1712
                                         0A      A2  ** 0002D          PUSHAB  10(SNK)                                         ;
                                       0000'     CF  9F 00030          PUSHAB  P.AAO                                           ;
                             00000000G   00      04  FB 00034          CALLS   #4, SYS$ASSIGN                                  ;
                                         2A      50  E9 0003B          BLBC    STATUS, 2$                                      ; 1714
                                         63      01  88 0003E          BISB2   #1, (R3)                                        ; 1717
                                 14      A2      0000V   CF  9E 00041  MOVAB   EVL$OPENDONE, 20(SNK)                          ; 1718
                                                 7E  7C 00047          CLRQ    -(SP)                                           ; 1728
                                                 7E  7C 00049          CLRQ    -(SP)                                           ;
                                         30      A2  9F 0004B          PUSHAB  48(SNK)                                         ;
                                                 7E  D4 0004E          CLRL    -(SP)                                           ;
                                                 52  DD 00050          PUSHL   SNK                                             ;
                                       0000V     CF  9F 00052          PUSHAB  EVL$ASTWORK                                     ;
                                         0C      A2  9F 00056          PUSHAB  12(SNK)                                         ;
                                                 32  DD 00059          PUSHL   #50                                             ;
                                 7E      0A      A2  3C 0005B          MOVZWL  10(SNK), -(SP)                                  ;
                                                 02  DD 0005F          PUSHL   #2                                              ;
                             00000000G   00      0C  FB 00061          CALLS   #12, SYS$QIO                                    ;
                                                 7E  D4 00068 2$:      CLRL    -(SP)                                           ; 1732
                                                 50  DD 0006A          PUSHL   STATUS                                          ;
                                 00000000G       8F  DD 0006C          PUSHL   #EVL$_OPNSNK                                    ;
                         FE48    CF              03  FB 00072          CALLS   #3, EVL$NETERROR                                ;
                                 0B              50  E8 00077          BLBS    R0, 3$                                          ;
                                 63              10  88 0007A          BISB2   #16, (R3)                                       ; 1735
                                                 52  DD 0007D          PUSHL   SNK                                             ; 1736
                         0000V   CF              01  FB 0007F          CALLS   #1, EVL$CLOSESNK                                ;
                                                     04 00084          RET                                                     ;
                                 63              02  88 00085 3$:      BISB2   #2, (R3)                                        ; 1739
                                                     04 00088 4$:      RET                                                     ; 1741

; Routine Size:  137 bytes,    Routine Base:  $CODE$ + 07ED
```

```
;  1762          1742  1 %SBTTL  'EVL$OPENDONE  Finish Up the Sink Open'
;  1763          1743  1 ROUTINE EVL$OPENDONE (SNKBLK) :NOVALUE =
;  1764          1744  1
;  1765          1745  1 !++
;  1766          1746  1 ! FUNCTIONAL DESCRIPTION:
;  1767          1747  1 !
;  1768          1748  1 !       This is the work routine which handles the completion of an
;  1769          1749  1 !       access function on a logical link to an event receiver on a
;  1770          1750  1 !       sink node.  If any error occurred, then wait for a while before
;  1771          1751  1 !       trying the open again.  All errors are logged by signalling them.
;  1772          1752  1 !       If no error occurred, then call the write event routine to
;  1773          1753  1 !       start writing events to the sink node if any have been queued.
;  1774          1754  1 !
;  1775          1755  1 ! FORMAL PARAMETERS:
;  1776          1756  1 !
;  1777          1757  1 !       SNKBLK          Address of the sink block
;  1778          1758  1 !
;  1779          1759  1 ! IMPLICIT INPUTS:
;  1780          1760  1 !
;  1781          1761  1 !       NONE
;  1782          1762  1 !
;  1783          1763  1 ! IMPLICIT OUTPUTS:
;  1784          1764  1 !
;  1785          1765  1 !       NONE
;  1786          1766  1 !
;  1787          1767  1 ! ROUTINE VALUE:
;  1788          1768  1 ! COMPLETION CODES:
;  1789          1769  1 !
;  1790          1770  1 !       NONE
;  1791          1771  1 !
;  1792          1772  1 ! SIDE EFFECTS:
;  1793          1773  1 !
;  1794          1774  1 !       NONE
;  1795          1775  1 !
;  1796          1776  1 !--
;  1797          1777  1
;  1798          1778  2     BEGIN
;  1799          1779  2
;  1800          1780  2     BIND
;  1801          1781  2         OPENWAITTIME =                              ! Time to wait before trying
;  1802          1782  2             UPLIT (-60*10*1000*1000, -1)            ! open again = 1 minute
;  1803          1783  2         ;
;  1804          1784  2
;  1805          1785  2     LOCAL
;  1806          1786  2         SNK     : REF BBLOCK,                       ! Local pointer to sink block
;  1807          1787  2         STATUS                                      ! Local status
;  1808          1788  2         ;
;  1809          1789  2
;  1810          1790  2     SNK = .SNKBLK;                                  ! Set the local pointer
;  1811          1791  2     IF NOT                                          ! If there was an error
;  1812          1792  2         EVL$NETERROR                                ! Report it and
;  1813          1793  2             (EVL$_OPNSNK, TRUE, SNK [SNK$W_IOSB] )
;  1814          1794  2     THEN
;  1815          1795  3         BEGIN
;  1816          1796  3         IF NOT
;  1817          1797  4             (STATUS = WKQ$ADD_TIMED_WORK            ! Schedule close to reopen
;  1818          1798  4                 (                                   ! sink later
```

```
    4                   EVL$CLOSESNK,                       ! Routine to activate
    4                   .SNK,                               ! Its parameters
    4                   0,                                  ! Dummy parm
    4                   OPENWAITTIME,                       ! Time to wait
    4                   .SNK                                ! Timer Request id
    4                   )
    4               )
    3       THEN
    3           SIGNAL (EVL$_WKQERR, 0, .STATUS)            ! Signal something wrong
    3       END
    2   ELSE
    3       BEGIN
    3       SNK [SNK$V_STS_BSY] = FALSE;                    ! Link is not busy
    3       EVL$WRITEVT (.SNK)                              ! No error, so write events
    3       END
    1   END;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                FFFFFFFF  DC3CBA00  00098 P.AAQ:        .LONG   -600000000, -1                          ;

                                        OPENWAITTIME=           P.AAQ


                                                        .PSECT  $CODE$,NOWRT,2

                                0004 00000 EVL$OPENDONE:
                                                        .WORD   Save R2                                 ; 1743
                52       04  AC  D0 00002               MOVL    SNKBLK, SNK                             ; 1790
                         0C  A2  9F 00006               PUSHAB  12(SNK)                                 ; 1793
                             01  DD 00009               PUSHL   #1                                      ;
                    00000000G 8F DD 0000B               PUSHL   #EVL$_OPNSNK                            ;
         FE20  CF            03  FB 00011               CALLS   #3, EVL$NETERROR                        ;
                28           50  E8 00016               BLBS    R0, 1$                                  ;
                             52  DD 00019               PUSHL   SNK                                     ; 1803
                        0000' CF 9F 0001B               PUSHAB  OPENWAITTIME                            ; 1798
                             7E  D4 0001F               CLRL    -(SP)                                   ;
                             52  DD 00021               PUSHL   SNK                                     ; 1800
                        0000V CF 9F 00023               PUSHAB  EVL$CLOSESNK                            ; 1798
         0000G  CF           05  FB 00027               CALLS   #5, WKQ$ADD_TIMED_WORK                  ;
                1D           50  E8 0002C               BLBS    STATUS, 2$                              ;
                             50  DD 0002F               PUSHL   STATUS                                  ; 1807
                             7E  D4 00031               CLRL    -(SP)                                   ;
                    00000000G 8F DD 00033               PUSHL   #EVL$_WKQERR                            ;
    00000000G  00            03  FB 00039               CALLS   #3, LIB$SIGNAL                          ;
                             04     00040               RET                                             ; 1795
         2E    A2            02  8A 00041 1$:           BICB2   #2, 46(SNK)                             ; 1811
                             52  DD 00045               PUSHL   SNK                                     ; 1812
         FE2B  CF            01  FB 00047               CALLS   #1, EVL$WRITEVT                         ;
                             04     0004C 2$:           RET                                             ; 1814

; Routine Size:  77 bytes,    Routine Base:  $CODE$ + 0876
```

```
;1836   1815  1 %SBTTL  'EVL$CLOSESNK  Close Link to a Sink Node'
;1837   1816  1 ROUTINE EVL$CLOSESNK (SNKBLK) :NOVALUE =
;1838   1817  1
;1839   1818  1 !++
;1840   1819  1 ! FUNCTIONAL DESCRIPTION:
;1841   1820  1 !
;1842   1821  1 !       Close the logical link to a sink block by deassigning the channel
;1843   1822  1 !       to net.  This causes an abort.  This is preferrable to a deaccess
;1844   1823  1 !       function to the netacp and is certainly much simpler for us.
;1845   1824  1 !       There are evidently theoretical problems associated with synchronous
;1846   1825  1 !       disconnects on logical links.  The abort is guaranteed to work
;1847   1826  1 !       correctly.  The outstanding read in the receiver gets an ABORT
;1848   1827  1 !       status on its read and simply breaks the link.
;1849   1828  1 !
;1850   1829  1 !       When the link is broken, if the sink block has DELete or CLoSe status
;1851   1830  1 !       set, deallocate the sinkblock since we are done with it.  Otherwise
;1852   1831  1 !       call writevt to reopen the link if any events are waiting.
;1853   1832  1 !       This mechanism allows us to temporarily close down links to sink nodes
;1854   1833  1 !       which are very seldom used.  If a link has not been used in some time,
;1855   1834  1 !       simply calling closesnk will close it until events are queued to it.
;1856   1835  1 !
;1857   1836  1 !       If the last sink block is closed, we set a flag to indicate we are not
;1858   1837  1 !       doing any work now.  The main line may elect to exit if nobody is
;1859   1838  1 !       doing any work.
;1860   1839  1 !
;1861   1840  1 ! FORMAL PARAMETERS:
;1862   1841  1 !
;1863   1842  1 !       SNKBLK          Address of the sink block
;1864   1843  1 !
;1865   1844  1 ! IMPLICIT INPUTS:
;1866   1845  1 !
;1867   1846  1 !       NONE
;1868   1847  1 !
;1869   1848  1 ! IMPLICIT OUTPUTS:
;1870   1849  1 !
;1871   1850  1 !       NONE
;1872   1851  1 !
;1873   1852  1 ! ROUTINE VALUE:
;1874   1853  1 ! COMPLETION CODES:
;1875   1854  1 !
;1876   1855  1 !       NONE
;1877   1856  1 !
;1878   1857  1 ! SIDE EFFECTS:
;1879   1858  1 !
;1880   1859  1 !       NONE
;1881   1860  1 !
;1882   1861  1 !--
;1883   1862  1
;1884   1863  2     BEGIN
;1885   1864  2
;1886   1865  2     LOCAL
;1887   1866  2         SNK     : REF BBLOCK,                   ! Pointer to sink block
;1888   1867  2         STATUS                                  ! Local status return
;1889   1868  2         ;
;1890   1869  2
;1891   1870  2     SNK = .SNKBLK;                              ! Set pointer to sink
;1892   1871  2     IF .SNK [SNK$V_STS_OPN]                     ! If sink is open
```

```
  2      THEN
  3          BEGIN                                   ! Deassign channel to net
P 3          STATUS = $DASSGN                        ! to close logical link
P 3              (
P 3              CHAN = .SNK [SNK$W_NETCHAN]         ! Channel is in snk block
  3              );
  3          EVL$NETERROR (EVL$_NETDAS, .STATUS, 0); ! Report but ignore error
  3          SNK [SNK$V_STS_OPN] = FALSE             ! Link is not open
  3          END
  2      ;
  2
  2 !
  2 !        The close could have been called by a timed work or for other
  2 !        reasons such as the database entry for the sink went away.
  2 !        If the timer bit is set, the timer may be running.  We cannot
  2 !        findout if it was the timer that called us or not, so we must
  2 !        try to cancel the timer entry.  If one is outstanding and we
  2 !        deallocate the sink block, bad news when it goes off.
  2 !
  2
  2      IF .SNK [SNK$V_STS_TMR]                     ! If the timer is running
  2      THEN                                        ! Cancel the work item even
  3          BEGIN
  3          WKQ$CANCEL_TIMED_WORK (.SNK);           ! if it is what called us.
  3          SNK [SNK$V_STS_TMR] = FALSE             ! Clear the bit
  3          END
  2      ;
  2
  2      IF  .SNK [SNK$V_STS_DEL]                    ! If link is to be deleted
  2          OR
  2          .SNK [SNK$V_STS_CLS]                    ! or closed
  2      THEN
  2          EVL$DEALLOCSNK (.SNK)                   ! Deallocate the sink block
  2      ELSE
  3          BEGIN
  3          SNK [SNK$V_STS_BSY] = FALSE;            ! Nothing active now
  3          EVL$WRITEVT (.SNK)                      ! Otherwise check for events
  3          END                                     ! and if so open link to write
  2      ;                                           ! them out
  1      END;


                                                 .EXTRN  SYS$DASSGN

                              000C 00000 EVL$CLOSESNK:
                                                 .WORD   Save R2,R3                 ; 1816
          52        04   AC   D0 00002           MOVL    SNKBLK, SNK                ; 1870
          53        2E   A2   9E 00006           MOVAB   46(SNK), R3                ; 1871
          1D             63   E9 0000A           BLBC    (R3), 1$                   ;
          7E        0A   A2   3C 0000D           MOVZWL  10(SNK), -(SP)             ; 1877
 00000000G  00           01   FB 00011           CALLS   #1, SYS$DASSGN             ;
                         7E   D4 00018           CLRL    -(SP)                      ; 1878
                         50   DD 0001A           PUSHL   STATUS                     ;
               00000000G 8F   DD 0001C           PUSHL   #EVL$_NETDAS               ;
     FDC2   CF           03   FB 00022           CALLS   #3, EVL$NETERROR           ;
```

```
                        63          01  8A 00027          BICB2   #1, (R3)                                ; 1879
                    0A  63          05  E1 0002A 1$:      BBC     #5, (R3), 2$                            ; 1892
                        63          52  DD 0002E          PUSHL   SNK                                     ; 1895
                 0000G  CF          01  FB 00030          CALLS   #1, WKQ$CANCEL_TIMED_WORK               ;
                        63          20  8A 00035          BICB2   #32, (R3)                               ; 1896
                    04  63          03  E0 00038 2$:      BBS     #3, (R3), 3$                            ; 1900
                    08  63          04  E1 0003C          BBC     #4, (R3), 4$                            ; 1902
                        63          52  DD 00040 3$:      PUSHL   SNK                                     ; 1904
                 F8EB   CF          01  FB 00042          CALLS   #1, EVL$DEALLOCSNK                      ;
                                        04 00047          RET                                             ;
                        63          02  8A 00048 4$:      BICB2   #2, (R3)                                ; 1907
                        63          52  DD 0004B          PUSHL   SNK                                     ; 1908
                 FDD8   CF          01  FB 0004D          CALLS   #1, EVL$WRITEVT                         ;
                                        04 00052          RET                                             ; 1912

; Routine Size:  83 bytes,    Routine Base:  $CODE$ + 08C3
```

```
;  1935          1913  1 %SBTTL  'EVL$ALLOCDBK  Allocate a Data Block'
;  1936          1914  1 GLOBAL ROUTINE EVL$ALLOCDBK (DBKSIZE, PRED, DBKRTN) =
;  1937          1915  1
;  1938          1916  1 !++
;  1939          1917  1 ! FUNCTIONAL DESCRIPTION:
;  1940          1918  1 !
;  1941          1919  1 !       Allocate a data block with a specified size and link it into a
;  1942          1920  1 !       queue in a specified place.  If there is an error, signal it and
;  1943          1921  1 !       then exit.
;  1944          1922  1 !
;  1945          1923  1 ! FORMAL PARAMETERS:
;  1946          1924  1 !
;  1947          1925  1 !       DBKSIZE         Size of data block
;  1948          1926  1 !       PRED            Predecessor in the queue
;  1949          1927  1 !       DBKRTN          Address to return address of allocated dbk
;  1950          1928  1 !
;  1951          1929  1 ! IMPLICIT INPUTS:
;  1952          1930  1 !
;  1953          1931  1 !       NONE
;  1954          1932  1 !
;  1955          1933  1 ! IMPLICIT OUTPUTS:
;  1956          1934  1 !
;  1957          1935  1 !       NONE
;  1958          1936  1 !
;  1959          1937  1 ! ROUTINE VALUE:
;  1960          1938  1 ! COMPLETION CODES:
;  1961          1939  1 !
;  1962          1940  1 !       Routine returns true if entry is first in the queue
;  1963          1941  1 !
;  1964          1942  1 ! SIDE EFFECTS:
;  1965          1943  1 !
;  1966          1944  1 !       NONE
;  1967          1945  1 !
;  1968          1946  1 !--
;  1969          1947  1
;  1970          1948  2     BEGIN
;  1971          1949  2
;  1972          1950  2     BUILTIN INSQUE ;
;  1973          1951  2
;  1974          1952  2     LOCAL
;  1975          1953  2         STATUS,                                 ! Hold on to status here
;  1976          1954  2         DBK : REF BBLOCK                        ! Local for the data block
;  1977          1955  2         ;
;  1978          1956  2
;  1979          1957  2     IF NOT
;  1980          1958  3         (STATUS = LIB$GET_VM (DBKSIZE, DBK) )   ! Obtain a block of data
;  1981          1959  2     THEN
;  1982          1960  3         BEGIN
;  1983          1961  3         SIGNAL (EVL$_INSFVM, 0, .STATUS);       ! Unable to obtain memory,
;  1984          1962  4         $EXIT (CODE = EVL$_INSFVM)               ! Its fatal so exit.
;  1985          1963  3         END
;  1986          1964  2     ;
;  1987          1965  2
;  1988          1966  2     CH$FILL (0, .DBKSIZE, .DBK);                ! Zero the area
;  1989          1967  2     DBK [DBK$W_SIZE] = .DBKSIZE;                ! Set the size in the block
;  1990          1968  2     .DBKRTN = .DBK;                             ! Return the address
;  1991          1969  2     INSQUE (.DBK, .PRED)                        ! Place on the queue
```

```
; 1992          1970  2
; 1993          1971  1     END;


                                                                .EXTRN  SYS$EXIT

                                                00FC 00000      .ENTRY  EVL$ALLOCDBK, Save R2,R3,R4,R5,R6,R7        ; 1914
                                    57 00000000G  8F   D0 00002 MOVL    #EVL$_INSFVM, R7                        ;
                                    5E            04   C2 00009 SUBL2   #4, SP                                  ;
                                                  5E   DD 0000C PUSHL   SP                                      ; 1958
                                              04  AC   9F 0000E PUSHAB  DBKSIZE                                 ;
                              00000000G     00    02   FB 00011 CALLS   #2, LIB$GET_VM                          ;
                                    16            50   E8 00018 BLBS    STATUS, 1$                              ;
                                                  50   DD 0001B PUSHL   STATUS                                  ; 1961
                                                  7E   D4 0001D CLRL    -(SP)                                   ;
                                                  57   DD 0001F PUSHL   R7                                      ;
                              00000000G     00    03   FB 00021 CALLS   #3, LIB$SIGNAL                          ;
                                                  57   DD 00028 PUSHL   R7                                      ; 1962
                              00000000G     00    01   FB 0002A CALLS   #1, SYS$EXIT                            ;
                                    56            6E   D0 00031 1$:     MOVL    DBK, R6                         ; 1966
      04    AC              00    6E            00   2C 00034 MOVC5   #0, (SP), #0, DBKSIZE, (R6)                ;
                                                  66       003A                                                 ;
                          08  A6        04  AC   B0 0003B MOVW    DBKSIZE, 8(R6)                                ; 1967
                          0C  BC            56   D0 00040 MOVL    R6, @DBKRTN                                   ; 1968
                                            50   D4 00044 CLRL    R0                                            ; 1969
                          08  BC            66   0E 00046 INSQUE  (R6), @PRED                                   ;
                                            02   12 0004A BNEQ    2$                                            ;
                                            50   D6 0004C INCL    R0                                            ;
                                                 04 0004E 2$:     RET                                           ; 1971

; Routine Size:  79 bytes,    Routine Base:  $CODE$ + 0916
```

```
;  1995          1972  1 %SBTTL  'EVL$DEALLOCDBK  Deallocate Data Block'
;  1996          1973  1 GLOBAL ROUTINE EVL$DEALLOCDBK (DBKADR) :NOVALUE =
;  1997          1974  1
;  1998          1975  1 !++
;  1999          1976  1 ! FUNCTIONAL DESCRIPTION:
;  2000          1977  1 !
;  2001          1978  1 !       Deallocate a data block.  This size is in the data block.
;  2002          1979  1 !       The data block is removed from the queue first.
;  2003          1980  1 !
;  2004          1981  1 ! FORMAL PARAMETERS:
;  2005          1982  1 !
;  2006          1983  1 !       DBKADR          Address of the dbk
;  2007          1984  1 !
;  2008          1985  1 ! IMPLICIT INPUTS:
;  2009          1986  1 !
;  2010          1987  1 !       NONE
;  2011          1988  1 !
;  2012          1989  1 ! IMPLICIT OUTPUTS:
;  2013          1990  1 !
;  2014          1991  1 !       NONE
;  2015          1992  1 !
;  2016          1993  1 ! ROUTINE VALUE:
;  2017          1994  1 ! COMPLETION CODES:
;  2018          1995  1 !
;  2019          1996  1 !       NONE
;  2020          1997  1 !
;  2021          1998  1 ! SIDE EFFECTS:
;  2022          1999  1 !
;  2023          2000  1 !       NONE
;  2024          2001  1 !
;  2025          2002  1 !--
;  2026          2003  2     BEGIN
;  2027          2004  2
;  2028          2005  2     BUILTIN REMQUE ;
;  2029          2006  2
;  2030          2007  2     LOCAL
;  2031          2008  2         SIZE,                       ! Size of the data block
;  2032          2009  2         DBK : REF BBLOCK            ! Local pointer to block
;  2033          2010  2         ;
;  2034          2011  2
;  2035          2012  2     REMQUE (.DBKADR, DBK);          ! Unlink the block
;  2036          2013  2     SIZE = .DBK [DBK$W_SIZE];       ! Save its size
;  2037          2014  2     LIB$FREE_VM (SIZE, DBK)         ! Free its memory
;  2038          2015  2
;  2039          2016  1     END;


                                    0000 00000          .ENTRY  EVL$DEALLOCDBK, Save nothing                ; 1973
                              5E    04  C2 00002          SUBL2   #4, SP                                      ;
                              7E    04  BC  0F 00005       REMQUE  @DBKADR, DBK                                ; 2012
                              50        6E  D0 00009       MOVL    DBK, R0                                     ; 2013
                        04    AE    08  A0  3C 0000C       MOVZWL  8(R0), SIZE                                 ;
                                        5E  DD 00011       PUSHL   SP                                          ; 2014
                                    08  AE  9F 00013       PUSHAB  SIZE                                        ;
```

```
                          00000000G  00          02  FB 00016          CALLS   #2, LIB$FREE_VM                          ;
                                                     04 0001D          RET                                             ; 2016

; Routine Size:  30 bytes,    Routine Base:  $CODE$ + 0965
```

```
;  2041          2017  1 %SBTTL  'EVL$ASTWORK  General AST Routine to Queue Work'
;  2042          2018  1 ROUTINE EVL$ASTWORK (ASP) :NOVALUE =
;  2043          2019  1
;  2044          2020  1 !++
;  2045          2021  1 ! FUNCTIONAL DESCRIPTION:
;  2046          2022  1 !
;  2047          2023  1 !       This routine is called as an AST routine on the completion of
;  2048          2024  1 !       an I/O request of some kind.  The parameter is an ASP block
;  2049          2025  1 !       or a SNK block (these have the same header format).  This block
;  2050          2026  1 !       has a routine address in the header which is the address of the
;  2051          2027  1 !       routine to queue as work.  The ASP block is passed as a parameter
;  2052          2028  1 !       to the routine.  No processing or error checking is done at AST
;  2053          2029  1 !       level.   This synchronizes all operations and avoids any possibility
;  2054          2030  1 !       of race conditions with AST routines.
;  2055          2031  1 !
;  2056          2032  1 ! FORMAL PARAMETERS:
;  2057          2033  1 !
;  2058          2034  1 !       ASP             Address of ASP block
;  2059          2035  1 !
;  2060          2036  1 ! IMPLICIT INPUTS:
;  2061          2037  1 !
;  2062          2038  1 !       NONE
;  2063          2039  1 !
;  2064          2040  1 ! IMPLICIT OUTPUTS:
;  2065          2041  1 !
;  2066          2042  1 !       NONE
;  2067          2043  1 !
;  2068          2044  1 ! ROUTINE VALUE:
;  2069          2045  1 ! COMPLETION CODES:
;  2070          2046  1 !
;  2071          2047  1 !       NONE
;  2072          2048  1 !
;  2073          2049  1 ! SIDE EFFECTS:
;  2074          2050  1 !
;  2075          2051  1 !       NONE
;  2076          2052  1 !
;  2077          2053  1 !--
;  2078          2054  1
;  2079          2055  2     BEGIN
;  2080          2056  2
;  2081          2057  2     MAP
;  2082          2058  2         ASP : REF BBLOCK
;  2083          2059  2         ;
;  2084          2060  2
;  2085          2061  2     LOCAL
;  2086          2062  2         STATUS
;  2087          2063  2         ;
;  2088          2064  2
;  2089          2065  2     IF .ASP [ASP$L_ROUTINE] LSSU 512
;  2090          2066  2     THEN
;  2091          2067  2         SIGNAL (EVL$_WKQERR, 0, SS$_ACCVIO)     ! Some error occurred
;  2092          2068  2     ;
;  2093          2069  2     IF NOT
;  2094          2070  3         ( STATUS = WKQ$ADD_WORK_ITEM    ! Add item to work queue
;  2095          2071  3             (
;  2096          2072  3             .ASP [ASP$L_ROUTINE],       ! Routine address
;  2097          2073  3             .ASP,                       ! First parameter
```

```
;  2098          2074  3              0                                ! Second parameter
;  2099          2075  3              )
;  2100          2076  3          )
;  2101          2077  2      THEN
;  2102          2078  2          SIGNAL (EVL$_WKQERR, 0, .STATUS) ! Some error occurred
;  2103          2079  2
;  2104          2080  1      END;


                                        001C 00000 EVL$ASTWORK:
                                                           .WORD   Save R2,R3,R4                                   ; 2018
                                 54 00000000G   8F   D0 00002      MOVL    #EVL$_WKQERR, R4
                                 53 00000000G   00   9E 00009      MOVAB   LIB$SIGNAL, R3
                                 52        04   AC   D0 00010      MOVL    ASP, R2                                 ; 2065
                          00000200    8F        14   A2   D1 00014      CMPL    20(R2), #512
                                                     09   1E 0001C      BGEQU   1$
                                                     0C   DD 0001E      PUSHL   #12                             ; 2067
                                                     7E   D4 00020      CLRL    -(SP)
                                                     54   DD 00022      PUSHL   R4
                                 63                  03   FB 00024      CALLS   #3, LIB$SIGNAL
                                                     7E   D4 00027 1$:  CLRL    -(SP)                           ; 2071
                                                     52   DD 00029      PUSHL   R2                              ; 2073
                                              14     A2   DD 0002B      PUSHL   20(R2)                          ; 2072
                              0000G   CF              03   FB 0002E      CALLS   #3, WKQ$ADD_WORK_ITEM
                                 09                  50   E8 00033      BLBS    STATUS, 2$                      ; 2071
                                                     50   DD 00036      PUSHL   STATUS                          ; 2078
                                                     7E   D4 00038      CLRL    -(SP)
                                                     54   DD 0003A      PUSHL   R4
                                 63                  03   FB 0003C      CALLS   #3, LIB$SIGNAL
                                                     04 0003F 2$:       RET                                     ; 2080

; Routine Size:  64 bytes,    Routine Base:  $CODE$ + 0983
```

```
; 2106          2081  1 END                           !End of module
; 2107          2082  0 ELUDOM


                                                      .EXTRN  LIB$SIGNAL, LIB$STOP

;                               PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $PLIT$                        160  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $GLOBAL$                       29  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $OWN$                           4  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                       2499  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                           Library Statistics
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1      9776        19         0         581        00:01.0
;  _$255$DUA28:[SHRLIB]NET.L32;1          1279         6         0          63        00:00.9
;  _$255$DUA28:[EVL.OBJ]EVLIBRARY.L32;1    191        53        27          14        00:00.2
;  _$255$DUA28:[EVL.OBJ]EVCDEF.L32;1       213        18         8          15        00:00.2


;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:EVLTRANS/OBJ=OBJ$:EVLTRANS MSRC$:EVLTRANS/UPDATE=(ENH$:EVLTRANS)

; Size:          2499 code + 193 data bytes
; Run Time:         00:47.1
; Elapsed Time:     01:47.8
; Lines/CPU Min:     2650
; Lexemes/CPU-Min: 18128
; Memory Used:  175 pages
; Compilation Complete
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

EVLJULIAN
LIS

EVCDEF
LIS

ERRMSG
LIS

EVLIBRARY
LIS

EVLTRANS
LIS

EVLSHOW
LIS

EVLMAIN
LIS

RECEIVER
LIS