M 8

```
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
```

FILEID**EVLTRANS

K 8

```
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
```

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

```
;   1 !               Convert to use new NETACP control QIO interface.
;   1 !--
```

```
1 %SBTTL  'Definitions'
1
1 !
1 ! TABLE OF CONTENTS:
1 !
1
1 FORWARD ROUTINE
1
1               EVL$TRANSMIT     : NOVALUE,         ! Initialize event transmitter
1               EVL$BLDFILTERS   : NOVALUE,         ! Build sink and filter database
1               EVL$BLDSOURCES   : NOVALUE,         ! Build source block on sink block
1               EVL$ALLOCSNK     : NOVALUE,         ! Allocate a sink block
1               EVL$DEALLOCSNK   : NOVALUE,         ! Deallocate a sink block
1               EVL$OBTAINEVTS   : NOVALUE,         ! Wait on events available from NETACP
1               EVL$READEVTS     : NOVALUE,         ! Read raw events from NETACP
1               EVL$PROCESSEVTS  : NOVALUE,         ! Process raw event buffer
1               EVL$FILTEREVT    : NOVALUE,         ! Filter a single raw event
1               EVL$FILTERSRC    :,                 ! Check filters with a matching source
1               EVL$QUEUEEVT     : NOVALUE,         ! Translate and queue an event to sink
1               EVL$BLDLOSTEVENT : NOVALUE,         ! Build and queue a lost event event
1               EVL$NETERROR     :,                 ! Report an error in network io
1               EVL$WRITEVT      : NOVALUE,         ! Write an event to sink
1               EVL$WRITEDONE    : NOVALUE,         ! Complete io to sink
1               EVL$OPENSNK      : NOVALUE,         ! Open a link to sink node
1               EVL$OPENDONE     : NOVALUE,         ! Complete open of link
1               EVL$CLOSESNK     : NOVALUE,         ! Close and possibly delete sink block
1               EVL$ALLOCDBK,                       ! Allocate a data block
1               EVL$DEALLOCDBK   : NOVALUE,         ! Deallocate a data block
1               EVL$ASTWORK      : NOVALUE          ! AST routine to queue work request
1               ;
1
1 !
1 ! INCLUDE FILES:
1 !
1
1 LIBRARY 'SYS$LIBRARY:STARLET';                    ! VMS common definitions
1 LIBRARY 'SHRLIB$:NET';                            ! Network ACP interface
1 LIBRARY 'LIB$:EVLIBRARY';                         ! EVL definitions
1 LIBRARY 'LIB$:EVCDEF';                            ! Event record definitions
1
1 !
1 ! EQUATED SYMBOLS:
1 !
1
1 LITERAL
1               EVL$C_EVTMBXSIZE = 64,                  ! Size of event mailbox
1               EVL$C_EVTBFRSIZE = 512 - ASP$C_SIZE     ! Size of event buffer
1               ;
1
1
1 BIND
1       EVL$Q_CLOSEDELAY =                              ! Delay before closing link
1               UPLIT (-120*10*1000*1000, -1)           ! to event receiver
1               ;
```

```
!
! OWN STORAGE:
!

GLOBAL
    EVL$GQ_SNKHEAD      : VECTOR [2]              ! Head of sink node list
                INITIAL (EVL$GQ_SNKHEAD, EVL$GQ_SNKHEAD),
    EVL$GT_LOCALNODE    : VECTOR [10, BYTE],      ! Buffer for local node info
    EVL$GB_LOCALNODE    : BYTE,                   ! Length of data in buffer
    EVL$GQ_ASPHEAD      : VECTOR [2]              ! Head of ASP queue
                INITIAL (EVL$GQ_ASPHEAD, EVL$GQ_ASPHEAD),
    EVL$GB_TRANSDONE    : BYTE INITIAL(TRUE)      ! True for transmitter done
    ;

OWN
    EVL$W_NETEVTCHAN    : WORD,                   ! Network channel for events
    EVL$W_MBXEVTCHAN    : WORD                    ! Mailbox channel for events
    ;
```

```
1
1 !
1 ! EXTERNAL REFERENCES:
1 !
1
1 EXTERNAL LITERAL
1
1         EVL$_NETASN,                ! Error assigning to NET
1         EVL$_OBTEVT,                ! Error obtaining raw events
1         EVL$_RAWFMT,                ! Invalid format for raw event
1         EVL$_WRTEVT,                ! Error writing events to sink node
1         EVL$_OPNSNK,                ! Error opening sink node link
1         EVL$_NETDAS,                ! Error deassigning NET channel
1         EVL$_INSFVM,                ! Unable to obtain virtual memory
1         EVL$_EVTSIZ,                ! Invalid size of event
1         EVL$_WKQERR,                ! Error returned from work queue routines
1         EVL$_LOGDBUT,               ! Log of data base updates by transmitter
1         EVL$_LOGOPNT,               ! Log of open links by transmitter
1         EVL$_NETSHUT                ! Network is shutting down
1         ;
1
1 EXTERNAL
1         EVL$GL_LOGMASK : BBLOCK ! Log control mask
1         ;
1
1 EXTERNAL ROUTINE
1
1         EVL$JULIAN              : NOVALUE,      ! Convert time to julian
1         WKQ$ADD_WORK_ITEM       : ,             ! Add element to work queue
1         WKQ$ADD_TIMED_WORK      : ,             ! Add scheduled work to queue
1         WKQ$CANCEL_TIMED_WORK   : ,             ! Cancel timed work item
1         EVL$QUEUE_EVENT         : NOVALUE,      ! Queue event to local receiver
1         EVL$NETSHOW             : ,             ! Show data from NETACP
1         EVL$PRINTLOG            : NOVALUE,      ! Print message to log
1         EVL$INITLOCALNODE       : NOVALUE       ! Initialize local node name
1         ;
1
1 !
1 !       Library routines
1 !
1
1 EXTERNAL ROUTINE
1
1         LIB$GET_VM : ADDRESSING_MODE (GENERAL),
1         LIB$FREE_VM : ADDRESSING_MODE (GENERAL),
1         LIB$ASN_WTH_MBX : ADDRESSING_MODE (GENERAL)
1         ;
```

```
1 %SBTTL  'EVL$TRANSMIT  Main Initialization'
1 GLOBAL ROUTINE EVL$TRANSMIT :NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Main initialization routine for event logging transmitter.
1 !       Calls the routines to initialize the data base here for the
1 !       transmitter and start the event logging.  After this routine
1 !       returns to its caller, work queue elements are on the work queue
1 !       to continue the event logging.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2     BEGIN
2
2     LOCAL
2         STATUS,                             ! Local status
2         IOSB     : VECTOR [2],              ! Iosb for simple qio
2         MBXMODES : VECTOR [2]               ! Enable mailbox modes
2         ;
2
2     EVL$INITLOCALNODE ();         ! Initialize the local node name
2
2     IF NOT
3         ( STATUS = LIB$ASN_WTH_MBX          ! Get some channels for use
3             (
3             %ASCID ' NET:',                 ! To network
3             0, %REF (EVL$C_EVTMBXSIZE),     ! Size of mailbox
3             EVL$W_NETEVTCHAN,               ! Network channel
3             EVL$W_MBXEVTCHAN                ! Mailbox channel
3             )
3         )
2     THEN
2         SIGNAL_STOP (EVL$_NETASN, 0, .STATUS)    ! Stop right now
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$_OBTEVT
                                                          .EXTRN  EVL$_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$GL_LOGMASK, EVL$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
```

```
1 %SBTTL  'EVL$BLDFILTERS  Build Our Local Database'
1 ROUTINE EVL$BLDFILTERS :NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Update the local database for the event transmitter.  If a sink node
1 !       is not present it is added to the queue.  All source filters on all
1 !       sinks are replaced with the latest data.  Sinks that are not present
1 !       are either deallocated outright, or if there are events in the queue,
1 !       marked for later close and deallocate.
1 !
1 !       If at the end of this routine, there are no sink blocks on the queue,
1 !       we set a flag saying we have no work to do.  Otherwise we set the
1 !       flag saying that we have work to do.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NETACP logging database
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       Sink node queue updated
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2     BEGIN
2
2     LITERAL
2         SNKBFRSIZE = 512                      ! Size of sink buffer
2         ;
2
2     LOCAL
2         SNKQUE,                               ! Pointer to sink queue head
2         POSITION:VECTOR [NFB$C_CTX_SIZE,BYTE],  ! Current EFI position
2         SNKBFR : BBLOCK [SNKBFRSIZE],         ! Buffer for sink data
2         SNKDSC : VECTOR [2],                  ! Descriptor of buffer
2         SNKPTR : REF BBLOCK;                  ! Pointer to sink block
2
2     SNKQUE = EVL$GQ_SNKHEAD;                  ! Head of sink queue
2     SNKPTR = ..SNKQUE;                        ! Setup pointer
2
2     WHILE .SNKPTR NEQ .SNKQUE                 ! Scan current queue
2     DO
```

```
                    BEGIN
                    SNKPTR [SNK$V_STS_DEL] = TRUE;   ! Set deleted bit in all
                    SNKPTR = .SNKPTR [SNK$L_FL]      ! Link to next
                    END
                ;

                POSITION <0,16> = 0;                 ! Start at first EFI record

                SNKDSC [0] = SNKBFRSIZE;             ! Descriptor of return buffer
                SNKDSC [1] = SNKBFR;

                WHILE                                ! As long as there are more nodes
                    EVL$NETSHOW(                     ! Obtain a sink node
                        NFB$C_DB_EFI,                ! Event information
                        NFB$C_WILDCARD, 0,           ! No search key
                        POSITION,                    ! Get next record, update position
                        2, UPLIT (NFB$C_EFI_SIN, NFB$C_EFI_EVE),
                        SNKDSC)                      ! Buffer descriptor
                DO
                    BEGIN
                    SNKPTR = ..SNKQUE;               ! Scan the queue for the new sink
                    IF
                        BEGIN
                        WHILE .SNKPTR NEQ .SNKQUE    ! Scan to the end
                        DO
                            BEGIN
                            IF .SNKBFR [0,0,32,0] EQL .SNKPTR [SNK$L_SNKADR]        ! Do the addresses
                            THEN                                                     ! match?
                                BEGIN
                                SNKPTR [SNK$V_STS_DEL] = FALSE;      ! Its here, so keep it
                                SNKPTR [SNK$V_STS_CLS] = FALSE;
                                EVL$BLDSOURCES (.SNKPTR, SNKBFR+4);  ! Build rest of data
                                EXITLOOP FALSE                       ! Do not create one
                                END
                            ;
                            SNKPTR = .SNKPTR [SNK$L_FL]              ! Link to next
                            END
                        END
                    THEN
                        BEGIN
                        EVL$ALLOCSNK (.SNKQUE, SNKPTR);              ! Allocate a new snk
                        SNKPTR [SNK$L_SNKADR] = .SNKBFR [0,0,32,0];  ! Save the sink addr
                        SNKPTR [SNK$L_SNKLEN] = SNK$S_SNKNCB;        ! Build ncb dsc
                        SNKPTR [SNK$A_SNKNCB] = SNKPTR [SNK$T_SNKNCB];
    P                   $FAOL                                        ! Build the ncb
    P                       (                                        ! in the sink block
    P                       CTRSTR = %ASCID '!UL::"26=/',            ! Use the address form
    P                       OUTLEN = SNKPTR [SNK$L_SNKLEN],          ! of the connect
    P                       OUTBUF = SNKPTR [SNK$L_SNKLEN],
    P                       PRMLST = SNKBFR
                            );
                        CH$MOVE(20, UPLIT BYTE(                      ! Append:
                                    0,0,                             ! Link number
                                    3, 4,0,0,                        ! Our version number
                                    REP 13 OF (0),                   ! Pad out unused data
                                    '"'),                            ! and NCB terminator
                            .SNKPTR [SNK$A_SNKNCB] + .SNKPTR [SNK$L_SNKLEN]);
```

```
;   0370 4                 SNKPTR [SNK$L_SNKLEN] = .SNKPTR [SNK$L_SNKLEN]+20; ! Adjust length
;   0371 4                 EVL$BLDSOURCES (.SNKPTR, SNKBFR+4)          ! and fill it in
;   0372 4                 END
;   0373 3             END
;   0374 2         ;
;   0375 2   !
;   0376 2   !
;   0377 2   !         Scan the snk list and remove deleted nodes
;   0378 2   !
;   0379 2
;   0380 2         SNKPTR = ..SNKQUE;
;   0381 2         WHILE .SNKPTR NEQ .SNKQUE                        ! Scan the list
;   0382 2         DO
;   0383 3             BEGIN
;   0384 3             IF .SNKPTR [SNK$V_STS_DEL]                   ! If deleted,
;   0385 3                 AND NOT .SNKPTR [SNK$V_STS_CLS]          ! and not already being closed,
;   0386 3             THEN
;   0387 4                 BEGIN
;   0388 4                 SNKPTR [SNK$V_STS_CLS] = TRUE;           ! Set close as true
;   0389 4                 WKQ$ADD_WORK_ITEM(EVL$WRITEVT,.SNKPTR); ! Make sure I/O going
;   0390 4                                                          ! (will delete if I/O done)
;   0391 4                 IF .EVL$GL_LOGMASK [ELG$V_DBUPDAT]       ! Log data base updates?
;   0392 4                 THEN
;   0393 4                     SIGNAL (EVL$_LOGDBUT, 1, 0)          ! Log that it occurred
;   0394 3                 END;
;   0395 3             SNKPTR = .SNKPTR [SNK$L_FL];                 ! Next snk in list
;   0396 2             END;
;   0397 2
;   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      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$BLDSOURCES       ;
                        FF7A 31 000B0       BRW     3$                       ; 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
```

```
1 %SBTTL  'EVL$BLDSOURCES  Build List of Sources in SNK Block'
1 ROUTINE EVL$BLDSOURCES (SNKBLK, SRCDATA) :NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Build source filter list on sink block.  All present source blocks
1 !       are removed first.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       SNKBLK          Address of sink block
1 !       SRCDATA         Address of source data list
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2     BEGIN
2
2     LOCAL
2         SNK : REF BBLOCK,                 ! Pointer to sink block
2         SRC : REF BBLOCK,                 ! Pointer to source block
2         SRCPTR,                           ! Pointer to source data
2         SRCSIZE,                          ! Size of a source block
2         SRCEND                            ! End of source data
2         ;
2
2     SNK = .SNKBLK;                        ! Deallocate all current sources
2     UNTIL SNK [SNK$L_SRCFL] EQL .SNK [SNK$L_SRCFL]
2     DO
2         EVL$DEALLOCDBK (.SNK [SNK$L_SRCFL] )
2     ;
2
2     SRCEND = .SRCDATA + .(.SRCDATA) <0, 16>; ! Word counted string
2     SRCPTR = .SRCDATA + 2;                ! Skip the count
2
2     WHILE .SRCPTR LSSU .SRCEND            ! For as long as there is data
2     DO
3         BEGIN
3         SRCSIZE = .(.SRCPTR) <0, 16>;     ! Size of source block
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
```

```
1 %SBTTL 'EVL$ALLOCSNK  Allocate and Initialize a SNK Block'
1 GLOBAL ROUTINE EVL$ALLOCSNK (PRED, SNKRTN) :NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Allocate and initialize a sink block.  The block is linked into the
1 !       correct place in the sink queue.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       PRED            Address of predecessor in list
1 !       SNKRTN          Address to return address of snk block
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2    BEGIN
2
2    LOCAL
2       SNK : REF BBLOCK
2       ;
2
2    EVL$ALLOCDBK (SNK$C_SIZE, .PRED, SNK);        ! Allocate the block
2    SNK [SNK$L_SRCFL] = SNK [SNK$L_SRCFL];        ! Fill in the queue headers
2    SNK [SNK$L_SRCBL] = SNK [SNK$L_SRCFL];
2    SNK [SNK$L_EVTFL] = SNK [SNK$L_EVTFL];
2    SNK [SNK$L_EVTBL] = SNK [SNK$L_EVTFL];
2    .SNKRTN = .SNK;                               ! Return the sink address
2
2    EVL$GB_TRANSDONE = FALSE;                     ! Mark transmitter active
2
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
```

```
1 %SBTTL  'EVL$DEALLOCSNK  Deallocate Sink Block'
1 ROUTINE EVL$DEALLOCSNK (SNKBLK) :NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !         Deallocate a sink block.  Remove and deallocate all the entries
1 !         on the src and evt queues as well.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !         SNKBLK          Address of the sink block
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !         NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !         NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !         NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !         NONE
1 !
1 !--
1
2     BEGIN
2
2     LOCAL
2         SNK : REF BBLOCK                            ! Local pointer to snk
2         ;
2
2     SNK = .SNKBLK;                                  ! Set the pointer
2     WHILE
2         .SNK [SNK$L_SRCFL]                          ! Watch for empty queue
2         NEQ
2         SNK [SNK$L_SRCFL]
2     DO
2         EVL$DEALLOCDBK (.SNK [SNK$L_SRCFL] )        ! Deallocate each source
2     ;
2
2     WHILE                                           ! Watch for end of queue
2         .SNK [SNK$L_EVTFL]
2         NEQ
2         SNK [SNK$L_EVTFL]
2     DO
2         EVL$DEALLOCDBK (.SNK [SNK$L_EVTFL] )        ! Deallocate each event
2     ;
2
2     EVL$DEALLOCDBK (.SNK);                          ! Deallocate snk block
2
```

```
;        0577  2      EVL$GB_TRANSDONE = (.EVL$GQ_SNKHEAD EQL EVL$GQ_SNKHEAD); ! Mark done if empty
;        0578  2
;        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
```

```
  1 %SBTTL  'EVL$OBTAINEVTS  Obtain Raw Events'
  1 ROUTINE EVL$OBTAINEVTS (ASPBLK) :NOVALUE =
  1
  1 !++
  1 ! FUNCTIONAL DESCRIPTION:
  1 !
  1 !       This routine reads a message from the mailbox associated with the
  1 !       net channel.  We complete the io in the readevts routine and
  1 !       then either try again for another mailbox message, or read
  1 !       events from the NETACP. If no aspblk is available, then we
  1 !       allocate one.
  1 !
  1 ! FORMAL PARAMETERS:
  1 !
  1 !       ASPBLK          Address of aspblk or zero if none
  1 !
  1 ! IMPLICIT INPUTS:
  1 !
  1 !       EVL$W_MBXEVTCHAN
  1 !
  1 ! IMPLICIT OUTPUTS:
  1 !
  1 !       NONE
  1 !
  1 ! ROUTINE VALUE:
  1 ! COMPLETION CODES:
  1 !
  1 !       NONE
  1 !
  1 ! SIDE EFFECTS:
  1 !
  1 !       NONE
  1 !
  1 !--
  1
  2     BEGIN
  2
  2     LOCAL
  2         ASP     : REF BBLOCK,                   ! Ast Paramter block
  2         STATUS
  2         ;
  2
  2     IF .ASPBLK EQL 0
  2     THEN
  2         EVL$ALLOCDBK                            ! Allocate a block if we need
  2             (                                   ! One
  2             EVL$C_EVTBFRSIZE + ASP$C_SIZE,
  2             .EVL$GQ_ASPHEAD [1],
  2             ASP
  2             )
  2     ELSE
  2         ASP = .ASPBLK                           ! Use old block if we have one
  2     ;
  2
  2     ASP [ASP$L_ROUTINE] = EVL$READEVTS;         ! Setup the routine address
  2
P 2     STATUS = $QIO                               ! Start a read from the
```

```
;  646       P 0637  2                (                                       ! mailbox to findout when
;  647       P 0638  2                CHAN = .EVL$W_MBXEVTCHAN,               ! events are available
;  648       P 0639  2                EFN = EVL$C_ASYNCH_EFN,
;  649       P 0640  2                FUNC = IO$_READVBLK,
;  650       P 0641  2                IOSB = ASP [ASP$W_IOSB],
;  651       P 0642  2                ASTADR = EVL$ASTWORK,
;  652       P 0643  2                ASTPRM = .ASP,
;  653       P 0644  2                P1 = ASP [ASP$T_DATA],
;  654       P 0645  2                P2 = EVL$C_EVTMBXSIZE                   ! Size of mailbox messages
;  655         0646  2                );
;  656         0647  2
;  657         0648  2            EVL$NETERROR (EVL$_OBTEVT, .STATUS, 0)      ! Analyse errors
;  658         0649  2
;  659         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
```

```
1 %SBTTL 'EVL$READEVTS  Read Events from NETACP'
1 ROUTINE EVL$READEVTS (ASPBLK) :NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Routine called to obtain a buffer of raw events from NETACP.
1 !       The input asp has a mailbox message in it which has been read
1 !       from a net channel mailbox.  The mailbox code is MSG$_EVTAVL for
1 !       events are available; MSG$_NETSHUT for the network is shutting down.
1 !       Other codes are ignored.
1 !       The events are read into an ASP buffer and then schedule work for
1 !       EVL$PROCESSEVTS.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       ASPBLK          Address of an asp block
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2   BEGIN
2
2   LOCAL
2       ASP     : REF BBLOCK,                       ! Ast Paramter block
2       NFB     : VECTOR [5, BYTE],                 ! Network function block
2       NFBDSC  : VECTOR [2],                       ! Descriptor of same
2       RSLDSC  : VECTOR [2],                       ! Descriptor of result buffer
2       STATUS
2       ;
2
2   ASP = .ASPBLK;                                  ! Use old block if we have one
2
2   IF NOT EVL$NETERROR                             ! Analyse the error
2       (
2       EVL$_OBTEVT,                                ! Unable to obtain events
2       TRUE,
2       ASP [ASP$W_IOSB]
2       )
2   THEN
3       BEGIN
```

```
        EVL$OBTAINEVTS (.ASP);                      ! Try again
        RETURN
        END
    ;

    SELECTONE CH$RCHAR (ASP [ASP$T_DATA] ) OF       ! Dispatch on message types
    SET

    [MSG$_EVTAVL] :                                 ! Just continue with work
        0;

    [MSG$_NETSHUT] :                                ! Network shut down happening
        BEGIN
        LOCAL SNKPTR: REF BBLOCK;
        SNKPTR = .EVL$GQ_SNKHEAD;
        WHILE .SNKPTR NEQ EVL$GQ_SNKHEAD            ! Scan the list
        DO
            BEGIN
            SNKPTR [SNK$V_STS_CLS] = TRUE;          ! Set close as true
            WKQ$ADD_WORK_ITEM(EVL$WRITEVT,.SNKPTR); ! Make sure I/O going
                                                    ! (will delete when I/O done)
            SNKPTR = .SNKPTR [SNK$L_FL];            ! Next snk in list
            END;
        RETURN;                                     ! Do not reissue mailbox read
        END;

    [OTHERWISE] :                                   ! Ignore the code and
        BEGIN
        EVL$OBTAINEVTS (.ASP);                      ! try again
        RETURN
        END;

    TES;

    ASP [ASP$L_ROUTINE] = EVL$PROCESSEVTS;          ! Setup the routine address

    NFBDSC [0] = 5;                                 ! Set up the nfb to readevents
    NFBDSC [1] = NFB;
    NFB [0] = NFB$C_READEVENT;
    RSLDSC [0] = EVL$C_EVTBFRSIZE;                  ! Set up the result buffer dsc
    RSLDSC [1] = ASP [ASP$T_DATA];
    STATUS = $QIOW                                  ! Perform the qio
        (
        EFN = EVL$C_SYNCH_EFN,
        CHAN = .EVL$W_NETEVTCHAN,                   ! The channel
        FUNC = IO$_ACPCONTROL,                      ! function
        IOSB = ASP [ASP$W_IOSB],
        P1 = NFBDSC,                                ! Network function block
        P3 = RSLDSC,                                ! Return result length here
        P4 = RSLDSC                                 ! Result buffer
        );

    ASP [ASP$W_IOSB1] = .RSLDSC [0];                ! *TEMP* Set length in IOSB

    IF NOT .STATUS                                  ! Report an error
    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   AE   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
```

```
%SBTTL  'EVL$PROCESSEVTS  Process Raw Events'
ROUTINE EVL$PROCESSEVTS (ASPBLK) :NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Process raw events and queue them to the various sink nodes.
!       If an error occurred on the read of raw events, signal it and
!       perform another request for raw events.  If no error, scan
!       the raw event buffer and filter each raw event in the buffer.
!
! FORMAL PARAMETERS:
!
!       ASPBLK          Address of the ASP block of raw events
!
! IMPLICIT INPUTS:
!
!       EVL$GQ_SNKHEAD  Head of sink list
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       NONE
!
!--

    BEGIN

    LOCAL
        ASP : REF BBLOCK,               ! Pointers to various blocks we need
        RAW : REF BBLOCK,
        SNK : REF BBLOCK,
        RAWPTR,                         ! Pointer to a raw event
        RAWEND,                         ! End of all of them
        RAWDSC : VECTOR [2]             ! Descriptor of raw event
        ;

    ASP = .ASPBLK;                      ! If some error then,
    IF NOT EVL$NETERROR (EVL$_OBTEVT, TRUE, ASP [ASP$W_IOSB] )
    THEN
        BEGIN
        EVL$OBTAINEVTS (.ASP);          ! Just try again for some events
        RETURN
        END
    ;

    RAWPTR = ASP [ASP$T_DATA];          ! Set start pointer and
    RAWEND = .RAWPTR + .ASP [ASP$W_IOSB1]; ! end pointer
```

```
2
2       WHILE .RAWPTR LSSA .RAWEND              ! Scan the whole list of raw events
2       DO
3           BEGIN
3           RAW = .RAWPTR;                      ! Point to this event
3           RAWPTR = .RAWPTR + .RAW [RAW$W_BYTES]; ! and the next one
3           IF .RAWPTR GTRA .RAWEND             ! Is this event too long??
3           THEN
4               BEGIN
4               SIGNAL (EVL$_RAWFMT);           ! Signal some error and
4               EXITLOOP                        ! then stop
4               END
3           ;
3
3           RAWDSC [0] = .RAW [RAW$W_BYTES];    ! Build descriptor of raw event
3           RAWDSC [1] = .RAW;
3           EVL$PRINTLOG                        ! Log the raw event if required
3               (
3               $BITPOSITION (ELG$V_RAWEVT),    ! Bit number of responsible bit
3               %ASCID 'Raw event',             ! Header text
3               0,                              ! No extra text
3               RAWDSC                          ! Descriptor of data
3               );
3
3           IF  .RAW [RAW$W_EVTCODE]            ! Data base change event?
3               EQL
3               EVC$C_VMS_DBC
3           THEN
3               EVL$BLDFILTERS ()               ! Just rebuild the filters
3           ELSE
4               BEGIN
4               SNK = .EVL$GQ_SNKHEAD [0];      ! Scan the sink queue and
4               WHILE .SNK NEQA EVL$GQ_SNKHEAD
4               DO
5                   BEGIN
5                   EVL$FILTEREVT (.RAW, .SNK);     ! filter each raw event
5                   SNK = .SNK [SNK$L_FL]   ! look at next sink
5                   END
4               END
3           END
2       ;
2
2       EVL$OBTAINEVTS (.ASP)                   ! Back for some raw events
2
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
```

```
1 %SBTTL  'EVL$FILTEREVT  Filter Events with a Sink'
1 ROUTINE EVL$FILTEREVT (RAWEVT, SNKBLK) :NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !          Filter a single event with a single sink.  If the event matches
1 !          for any of the sources or the global filters, then its translated and
1 !          queued for this sink.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !          RAWEVT              Address of the raw event in the buffer
1 !          SNKBLK              Address of the sink control block
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !          NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !          NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !          NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !          NONE
1 !
1 !--
1
2     BEGIN
2
2     LOCAL
2          RAW : REF BBLOCK,                 ! Pointer to a raw event
2          SNK : REF BBLOCK,                 ! Pointer to the sink block
2          SRC : REF BBLOCK,                 ! Pointer to a source descriptor .
2          SINKMASK : BYTE,                  ! Sink mask (console, file, monitor)
2          SRCFOUND : BYTE                   ! Bool for source found
2          ;
2
2     RAW = .RAWEVT;                                        ! Set pointer for raw event
2     SNK = .SNKBLK;                                        ! Set pointer to sink block
2     SINKMASK = 0;                                         ! No sinks indicated yet
2     SRCFOUND = FALSE;                                     ! Specific filter not found
2
2     SRC = .SNK [SNK$L_SRCFL];                             ! Scan the source dsc from
2     WHILE .SRC NEQ SNK [SNK$L_SRCFL]                      ! the head of the list
2     DO
3          BEGIN
3          IF   .RAW [RAW$B_SRCTYP]                         ! If source types match
3               EQL
3               .SRC [SRC$B_SRCTYP]
3               AND
```

```
                    BEGIN                                   ! and the sources themselves
                    SELECTONEU .RAW [RAW$B_SRCTYP] OF       ! match according to the type
                    SET
                    [EVC$C_SRC_NON] : TRUE ;                ! No source type, always match
                    [EVC$C_SRC_LIN,EVC$C_SRC_CIR,           ! Compare ASCIC strings
                        EVC$C_SRC_MOD] :
                        BEGIN
                        CH$EQL
                            (
                            CH$RCHAR (SRC [SRC$T_SRCID] ), ! Line in the source block
                            SRC [SRC$T_SRCID] + T,         ! size and address
                            CH$RCHAR (RAW [RAW$T_SRCID]),  ! Counted string
                            RAW [RAW$T_SRCID] + T,         ! from the raw event
                            0                              ! fill
                            )
                        END;
                    [EVC$C_SRC_NOD] :                       ! Compare node id's
                        BEGIN
                        .( RAW [RAW$T_SRCID] ) <0, 16>      ! Compare only the node
                        EQL                                 ! addresses
                        .( SRC [SRC$T_SRCID] ) <0, 16>
                        END;

                    [EVC$C_SRC_ARE] :                       ! Compare area numbers
                        (CH$RCHAR(RAW [RAW$T_SRCID]) EQL CH$RCHAR(SRC [SRC$T_SRCID]));
                    TES
                    END
                THEN
                    BEGIN
                    SRCFOUND = TRUE;                        ! They match, so we found it
                    SINKMASK = .SINKMASK OR                 ! Add its sink contribution
                        EVL$FILTERSRC (.RAW, .SRC)          ! to the growing mask
                    END
                ;
                SRC = .SRC [SRC$L_FL]                       ! Link to the next source dsc
                END
            ;

            IF NOT .SRCFOUND                                ! If we did not find a source
            THEN
                BEGIN
                SRC = .SNK [SNK$L_SRCFL];                   ! Scan the source list
                WHILE .SRC NEQ SNK [SNK$L_SRCFL]            ! and look for global filters
                DO
                    BEGIN
                    IF  .SRC [SRC$B_SRCTYP]                 ! Consider only global filters
                        EQL
                        EVC$C_SRC_NON
                    THEN
                        BEGIN
                        SINKMASK = .SINKMASK OR             ! Combine its contribution
                            EVL$FILTERSRC (.RAW, .SRC)
                        END
                    ;
                    SRC = .SRC [SRC$L_FL]                   ! And look at remainder of
                    END                                     ! source descriptors
                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$QUEUEEVT       ;
                                   04 000AB 12$:    RET                            ; 0993

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

```
1 %SBTTL  'EVL$FILTERSRC  Filter an Event with one Source'
1 ROUTINE EVL$FILTERSRC (RAWEVT, SRCBLK) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Scan the filters with one source block and if the raw event is
1 !       passed by the filters, return the sink mask value as the value
1 !       of the routine.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       RAWEVT          Address of the raw event
1 !       SRCBLK          Address of the source descriptor block
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       If the event is passed by the filters, return the sinkmask from the
1 !       source.  Otherwise return zero.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2     BEGIN
2
2     LOCAL
2         RAW     : REF BBLOCK,                       ! Pointer to raw event
2         SRC     : REF BBLOCK,                       ! Pointer to source descriptor
2         FLT     : REF BBLOCK,                       ! Pointer to filter
2         CLASS,                                      ! Class from raw event
2         TYPE                                        ! Type number from raw event
2         ;
2
2     RAW = .RAWEVT;                                  ! Setup useful pointers
2     SRC = .SRCBLK;
2     FLT = SRC [SRC$T_FILTERS];                      ! First filter is here
2     CLASS = .RAW [RAW$V_EVTCLS];                    ! Event code from raw event
2     TYPE = .RAW [RAW$V_EVTTYP];                     ! Obtain type number
2
2     DECRU FLTS FROM .SRC [SRC$W_FILTERS] TO 1       ! Look at all the filters
2     DO
3         BEGIN
3         IF                                          ! Compare according to filter
4             BEGIN                                   ! class and wild codes
4             SELECTONE .FLT [FLT$V_WLDCOD] OF
```

```
; 1064      1051 4              SET
; 1065      1052 4              [EVC$C_WLDCLS_KNO] :                          ! Known events
; 1066      1053 4                  TRUE;
; 1067      1054 4              [EVC$C_WLDCLS_ALL] :                          ! All types of a class
; 1068      1055 5                  BEGIN
; 1069      1056 5                  IF .CLASS EQL .FLT [FLT$V_CLASS]
; 1070      1057 5                  THEN TRUE
; 1071      1058 5                  ELSE FALSE
; 1072      1059 4                  END;
; 1073      1060 4              [0] :                                         ! Class and type must match
; 1074      1061 5                  BEGIN
; 1075      1062 5                  IF .CLASS EQL .FLT [FLT$V_CLASS]
; 1076      1063 5                      AND                                   ! Check both type masks
; 1077      1064 5                      .BBLOCK [FLT [FLT$Q_TYPESLOG], 0, .TYPE, 1, 0]
; 1078      1065 5                      AND
; 1079      1066 5                      NOT .BBLOCK [FLT [FLT$Q_TYPESFIL], 0, .TYPE, 1, 0]
; 1080      1067 5                  THEN TRUE
; 1081      1068 5                  ELSE FALSE
; 1082      1069 4                  END;
; 1083      1070 4              TES
; 1084      1071 4              END
; 1085      1072 3          THEN RETURN 1^(.SRC [SRC$B_SNKTYPE]-1); ! Return the sink mask
; 1086      1073 3          FLT = .FLT + FLT$C_SIZE                   ! Next filter
; 1087      1074 3          END
; 1088      1075 2      ;
; 1089      1076 2      RETURN 0                                      ! No sink found
; 1090      1077 2
; 1091      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$:  ADDL2   #20, FLT            ; 1073
                    53  D7 00060      DECL    FLTS                ;
                    BC  12 00062 5$:  BNEQ    1$                  ;
                    50  D4 00064      CLRL    R0                  ; 1076
                        04 00066      RET                         ; 1078

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

```
%SBTTL  'EVL$QUEUEVT  Translate and Queue an Event'
ROUTINE EVL$QUEUEVT (RAWEVT, SNKBLK, SNKMASK) :NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Translate the raw event to DNA standard form, queue it to the
!       sink block and call the EVL$WRITEVT routine to start the
!       output process to the sink node.
!
! FORMAL PARAMETERS:
!
!       RAWEVT          Address of the raw event
!       SNKBLK          Address of the sink block
!       SNKMASK         Value of the sink mask to use
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       NONE
!
!--

    BEGIN

    LITERAL
        EVTSIZE = 300                           ! Size of temp event buffer
        ;

    LOCAL
        EVTBFR  : VECTOR [EVTSIZE, BYTE],       ! Place to build DNA event
        RAW     : REF BBLOCK,                   ! Pointer to raw event
        SNK     : REF BBLOCK,                   ! Pointer to sink block
        EVQ     : REF BBLOCK,                   ! Pointer to event block
        PTR,                                    ! Pointer to build event
        PTRADR,                                 ! Pointer in memory for parm
        SIZE,                                   ! Size of event
        EVTDSC  : VECTOR [2],                   ! Descriptor of event
        DATAPTR,                                ! Pointer to data of event
        HALFDAY,                                ! Julian half days
        SECS,                                   ! Seconds in half day
        MSECS                                   ! Milliseconds in second
        ;

    RAW = .RAWEVT;                              ! Set pointer to raw event
```

```
PTR = EVTBFR;                                      ! Pointer to build DNA event
SNK = .SNKBLK;                                     ! Point to the sink

IF .SNK [SNK$W_EVTCNT] GTRU EVL$C_MAXEVTCNT ! Event queue too long?
THEN
    BEGIN
    IF (                                           ! If sink has not been lost
        NOT .SNK [SNK$B_SNKLOS]                    ! of this type yet
        AND .SNKMASK
        )
        NEQ 0
    THEN
        BEGIN
        EVL$BLDLOSTEVENT                           ! Build lost event message
            (                                      ! For the correct sinks
            .SNKMASK AND NOT .SNK [SNK$B_SNKLOS],
            .SNK [SNK$L_EVTBL],                    ! Link in at tail of queue
            EVQ                                    ! Return address here
            );
        SNK [SNK$B_SNKLOS] =                       ! Remember the sinks we lost
            .SNK [SNK$B_SNKLOS] OR .SNKMASK;       ! So only one message
        SNK [SNK$W_EVTCNT] =                       ! Count another event
            .SNK [SNK$W_EVTCNT] + 1;
        EVTDSC [0] = .EVQ [EVQ$W_EVTSIZE];         ! Build descriptor of event
        EVTDSC [1] = EVQ [EVQ$T_EVENT];
        EVL$PRINTLOG                               ! Log event if requested
            (
            $BITPOSITION (ELG$V_QUEEVT),
            %ASCID 'Events Lost',
            SNK [SNK$L_SNKLEN],
            EVTDSC
            )
        END
    ;
    RETURN                                         ! We lost the event, so done
    END
ELSE
    SNK [SNK$B_SNKLOS] = FALSE                     ! No events lost now
;

IF .RAW [RAW$W_BYTES]                              ! Quick check of raw size
        GTRU EVTSIZE - (-27+37)                    ! to see if it fits in buffer
    OR .RAW [RAW$W_BYTES]                          ! Or if too small
        LSSU $BYTEOFFSET(RAW$T_DATA)
THEN
    BEGIN
    SIGNAL (EVL$_EVTSIZ);                          ! Signal some thing bad wrong
    RETURN                                         ! and ignore event
    END
;

EVL$JULIAN                                         ! Convert raw time to DNA
    (                                              ! format
    RAW [RAW$T_SYSTIM],                            ! 64 bit systime
    HALFDAY, SECS, MSECS                           ! Julian halfdays, seconds, ..
    );
```

```
2
2 !
2 !       Build the DNA format event in the buffer
2 !
2
2       CH$WCHAR_A (1, PTR);                            ! Just a code byte for event
2       CH$WCHAR_A (.SNKMASK, PTR);                     ! The sink mask byte
2       PTR = CH$COPY                                   ! Build more of the header
2           (
2           2, RAW [RAW$W_EVTCODE],                     ! Event code
2           2, HALFDAY,                                 ! Date and time in dna format
2           2, SECS,
2           2, MSECS,
2           0,                                          ! Just a dummy fill char
2           8, .PTR                                     ! Place to put it
2           );
2
2       PTR = CH$MOVE                                   ! Put in the source node
2           (                                           ! in node id format
2           .EVL$GB_LOCALNODE,                          ! from a special buffer
2           EVL$GT_LOCALNODE,                           ! Built in advance
2           .PTR
2           );
2
2       CH$WCHAR_A (.RAW [RAW$B_SRCTYP], PTR);          ! Copy the source type
2
2       SELECTONEU .RAW [RAW$B_SRCTYP] OF               ! and id depending on type
2       SET
2       [EVC$C_SRC_NON] : (0);                          ! Nothing for no source id
2       [EVC$C_SRC_LIN,EVC$C_SRC_CIR,                   ! Copy ASCIC string
2           EVC$C_SRC_MOD] :
3           BEGIN
3           PTR = CH$MOVE                               ! Line id is a counted string
3               (
3               CH$RCHAR (RAW [RAW$T_SRCID]) + 1,
3               RAW [RAW$T_SRCID],
3               .PTR
3               )
2           END;
2       [EVC$C_SRC_NOD] :                               ! Copy node id from raw
3           BEGIN                                       ! its word (adr) ascic (name)
3           PTR = CH$MOVE
3               (
3               CH$RCHAR (RAW [RAW$T_SRCID] + 2) + 3, ! Count and address with str
3               RAW [RAW$T_SRCID],
3               .PTR
3               )
2           END;
2
2       [EVC$C_SRC_ARE]:                                ! Copy area number
3           BEGIN
3           CH$WCHAR_A (CH$RCHAR(RAW [RAW$T_SRCID]), PTR);  ! Copy the area number
2           END;
2       TES;
2
2 !
2 !       Copy the associated data with the event.
```

```
!

    PTR = CH$MOVE(
            .RAW [RAW$W_BYTES] - $BYTEOFFSET(RAW$T_DATA),
            RAW [RAW$T_DATA],
            .PTR);

!
!       Compute size of event in dna format, allocate a dbk for it
!       and copy it to the block.  The dbk is queued at the end of the
!       list in sink block for transmission.
!

    SIZE = .PTR - EVTBFR;                               ! Size of event
    EVL$ALLOCDBK                                        ! Allocate a buffer
        (
        .SIZE + EVQ$C_SIZE,                             ! size of buffer
        .SNK [SNK$L_EVTBL],                             ! Queue at end of queue
        EVQ                                             ! Return address here
        );
    EVQ [EVQ$W_EVTSIZE] = .SIZE;                        ! Set its size
    CH$MOVE                                             ! Copy the event to buffer
        (
        .SIZE,
        EVTBFR,
        EVQ [EVQ$T_EVENT]
        );
    SNK [SNK$W_EVTCNT] =                                ! Count another event
        .SNK [SNK$W_EVTCNT] + 1;

    EVTDSC [0] = .EVQ [EVQ$W_EVTSIZE];                  ! Build descriptor of event
    EVTDSC [1] = EVQ [EVQ$T_EVENT];
    EVL$PRINTLOG                                        ! Log event if requested
        (
        $BITPOSITION (ELG$V_QUEEVT),                    ! Bit controlling logging
        %ASCID 'Queued event',                          ! Header text
        SNK [SNK$L_SNKLEN],                             ! Ncb is extra info
        EVTDSC                                          ! Descriptor of event
        );

    EVL$WRITEVT (.SNK)                                  ! Start the output if needed

    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$QUEUEEVT:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7                  ; 1080
              5E      FEBC   CE   9E 00002              MOVAB   -324(SP), SP
              57        04   AC   D0 00007              MOVL    RAWEVT, RAW                             ; 1135
              53        18   AE   9E 0000B              MOVAB   EVTBFR, PTR                             ; 1136
              56        08   AC   D0 0000F              MOVL    SNKBLK, SNK                             ; 1137
        00C8  8F        2C   A6   B1 00013              CMPW    44(SNK), #200                           ; 1139
                             49   1B 00019              BLEQU   2$
              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
```

```
%SBTTL 'EVL$BLDLOSTEVENT  Build a Lost Event Event'
ROUTINE EVL$BLDLOSTEVENT (SNKMASK, PRED, EVQRTN) :NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Build a lost event event and queue it to a sink block.  The sink
!       mask for the lost event is passed here.  The current system time
!       is included in the event.
!
! FORMAL PARAMETERS:
!
!       SNKMASK         Value of sink mask for event
!       PRED            Predecessor for insert into queue
!       EVQRTN          Address to return address of event block
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       NONE
!
!--

    BEGIN

    LOCAL
        EVQ     : REF BBLOCK,                   ! Pointer to event block
        PTR,                                    ! Pointer into event block
        SYSTIM  : VECTOR [2],                   ! 64 bit system time (now)
        HALFDAY,                                ! Julian half day for now
        SECS,                                   ! and the rest
        MSECS                                   ! of the dna time
        ;

    EVL$ALLOCDBK                                ! Obtain an event block
        (
        EVQ$C_SIZE + 22,                        ! Thats big enough
        .PRED,                                  ! In correct queue
        EVQ                                     ! Return address here
        );

    $GETTIM                                     ! When is now
        (
        TIMADR = SYSTIM
        );
```

```
; 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 (.SNKMASK, 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
```

```
%SBTTL  'EVL$NETERROR  Check and Report a Network Error'
ROUTINE EVL$NETERROR (CODE, STATUSVAL, IOSBADR) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Check the status value and/or the iosb status code of
!       an io request to the network.  If anything is amiss then
!       signal the error to the log and return failure.  If all is
!       well, return success.
!
! FORMAL PARAMETERS:
!
!       CODE            Value of the EVL condition code to signal
!       STATUSVAL       Value of status from a network request
!       IOSBADR         Address of the iosb of a network request
!                       a value of zero implies do not check the iosb
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       Success if no error, failure otherwise.
!       Any errors are signaled so they will be logged.
!
! SIDE EFFECTS:
!
!       NONE
!
!--

    BEGIN

    MAP
        IOSBADR : REF BBLOCK                    ! Its the address of a block
        ;

    LOCAL
        IOSBSTS                                 ! Hold its value here
        ;

    IF NOT .STATUSVAL                           ! Signal any error here
    THEN
        SIGNAL (.CODE, 0, .STATUSVAL)
    ;

    IOSBSTS =                                   ! Obtain the value
        BEGIN
        IF .IOSBADR EQL 0                       ! Unless there is no block
        THEN
            SUCCESS                             ! in which case use success
```

```
3          ELSE
3              .IOSBADR [0, 0, 16, 0]                ! Status value of iosb
2          END;
2
2      IF NOT .IOSBSTS                               ! If its bad, signal it
2      THEN
2          SIGNAL (.CODE, 0, .IOSBSTS)
2      ;
2
2      RETURN .STATUSVAL AND .IOSBSTS AND SUCCESS   ! and return problems
2
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

```
%SBTTL  'EVL$WRITEVT  Write an Event to a Sink Node'
ROUTINE EVL$WRITEVT (SNKBLK) :NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       If a write is in progress, just return from here.
!       If the event queue is empty, close and delete the sink
!       if the status bits are set, but in any case return from here.
!       If the sink is not open, open it.  We will return here when
!       the open completes.
!       If everything is go, then start the write on the open logical
!       link to the event receiver and check for any errors on the qio.
!       The remainder of the write is handled in the WRITEDONE routine
!       which is activated as work when the AST completes.
!
!       When we find the event queue empty, we set a timer which
!       will activate the close routine.  The link will be reopened
!       when an event is requeued.  If an event is written we cancel the
!       timer entry.
!
! FORMAL PARAMETERS:
!
!       SNKBLK          Address of the sink block
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       NONE
!
!--

    BEGIN

    LOCAL
        SNK     : REF BBLOCK,                   ! Pointer to sink block
        EVQ     : REF BBLOCK,                   ! Pointer to event queue block
        STATUS                                  ! Local status value
        ;

    SNK = .SNKBLK;                              ! Point to sink block

    IF .SNK [SNK$V_STS_BSY]                     ! If some action in progress
    THEN
        RETURN                                  ! Go away and wait for done
```

```
2       ;
2
2       IF   .SNK [SNK$L_EVTFL]                        ! If event list is empty
2            EQL
2            SNK [SNK$L_EVTFL]
2       THEN
3           BEGIN
3           IF   .SNK [SNK$V_STS_DEL]                  ! If delete or close set
3                OR
3                .SNK [SNK$V_STS_CLS]
3           THEN
3               EVL$CLOSESNK (.SNK)                    ! Close and deallocate sink
3           ELSE
4               BEGIN
4               SNK [SNK$V_STS_TMR] = TRUE;            ! Set bit for timer active
4               WKQ$ADD_TIMED_WORK                     ! Set work to perform close
4                   (
4                   EVL$CLOSESNK,                      ! Close after the delay
4                   .SNK, 0,                           ! Sink is the parameter
4                   EVL$Q_CLOSEDELAY,                  ! Time delay
4                   .SNK                               ! Request id is sink address
4                   )
4               END
3           ;
3           RETURN                                     ! and we are done here
3           END
2       ;
2
2       IF .SNK [SNK$V_STS_TMR]                        ! If close timer is active
2       THEN
3           BEGIN
3           WKQ$CANCEL_TIMED_WORK (.SNK);              ! Cancel the work element
3           SNK [SNK$V_STS_TMR] = FALSE                ! Clear the bit
3           END
2       ;
2
2       EVQ = .SNK [SNK$L_EVTFL];                      ! Point to the first event
2       SNK [SNK$L_ROUTINE] = EVL$WRITEDONE;           ! Setup the routine to go to
2
2       IF .SNK [SNK$L_SNKADR] EQL .EVL$GT_LOCALNODE<0,16>  ! If to local receiver,
2       THEN
3           BEGIN
3           SNK [SNK$V_STS_BSY] = TRUE;                ! Write is in progress
3           WKQ$ADD_WORK_ITEM(EVL$QUEUE_EVENT,         ! Queue event to local receiver
3               .EVQ [EVQ$W_EVTSIZE], EVQ [EVQ$T_EVENT]);
3           SNK [SNK$W_IOSB] = TRUE;                   ! Set write successful
3           EVL$ASTWORK(.SNK);                         ! and simulate I/O completion
3           RETURN;
2           END;
2
2       IF NOT .SNK [SNK$V_STS_OPN]                    ! If sink is not open
2       THEN
3           BEGIN
3           EVL$OPENSNK (.SNK);                        ! Open the sink node link
3           RETURN                                     ! we will come back here later
2           END;
2
```

```
; 1584      P 1567  2           STATUS = $QIO                                          ! Do the write
; 1585      P 1568  2               (
; 1586      P 1569  2               EFN = EVL$C_ASYNCH_EFN,
; 1587      P 1570  2               CHAN = .SNK [SNK$W_NETCHAN],                       ! Channel in sink block
; 1588      P 1571  2               FUNC = IO$_WRITEVBLK,                              ! Write data to link
; 1589      P 1572  2               IOSB = SNK [SNK$W_IOSB],                           ! Leave status here
; 1590      P 1573  2               ASTADR = EVL$ASTWORK,                              ! Our general ast routine
; 1591      P 1574  2               ASTPRM = .SNK,                                     ! Sink block is parameter
; 1592      P 1575  2               P1 = EVQ [EVQ$T_EVENT],                            ! Address of event
; 1593      P 1576  2               P2 = .EVQ [EVQ$Q_EVTSIZE]                          ! Size of event in bytes
; 1594        1577  2               );
; 1595        1578  2
; 1596        1579  2           IF NOT EVL$NETERROR (EVL$_WRTEVT, .STATUS, 0) ! Report any error
; 1597        1580  2           THEN
; 1598        1581  2               EVL$CLOSESNK (.SNK)                                ! Close the sink on an error
; 1599        1582  2           ELSE
; 1600        1583  2               SNK [SNK$V_STS_BSY] = TRUE                         ! Write is in progress
; 1601        1584  2
; 1602        1585  1           END;
```

```
                                          001C 00000 EVL$WRITEEVT:
                                                         .WORD   Save R2,R3,R4                                  ; 1454
                     52         04  AC  D0 00002         MOVL    SNKBLK, SNK                                    ; 1505
                     54         2E  A2  9E 00006         MOVAB   46(SNK), R4                                    ; 1507
          01         64             01  E1 0000A         BBC     #1, (R4), 1$                                   ;
                                        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$ASTWORK
                               0C     A2  9F 00098        PUSHAB  12(SNK)
                                      30  DD 0009B        PUSHL   #48
               7E              0A     A2  3C 0009D        MOVZWL  10(SNK), -(SP)
                                      02  DD 000A1        PUSHL   #2
        00000000G   00                0C  FB 000A3        CALLS   #12, SYS$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
```

```
1 %SBTTL  'EVL$WRITEDONE  Finish Up an Event Write'
1 ROUTINE EVL$WRITEDONE (SNKBLK) :NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine is queued as work by the ast completion of
1 !       a write of an event to a sink.  The parameter is the sink block.
1 !       Check for an error on the write and if there was none, then
1 !       dequeue the event.  If there was an error, close the sink.
1 !       This will cause the sink to be reopened, creating a new logical
1 !       link and a retransmission will occur.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       SNKBLK          Address of the sink block of concern
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2     BEGIN
2
2     LOCAL
2         SNK     : REF BBLOCK                      ! Pointer to the sink block
2         ;
2
2     SNK = .SNKBLK;                                ! Set the pointer
2     SNK [SNK$V_STS_BSY] = FALSE;                  ! Write is not in progress
2
2     IF EVL$NETERROR                               ! Check for an error
2         (EVL$_WRTEVT, TRUE, SNK [SNK$W_IOSB] )
2     THEN
3         BEGIN
3         EVL$DEALLOCDBK (.SNK [SNK$L_EVTFL] );     ! No error, dump that event
3         SNK [SNK$W_EVTCNT] =                      ! Reduce count of queued evts
3             .SNK [SNK$W_EVTCNT] - 1;
3         EVL$WRITEVT (.SNK)                        ! and write the next one
3         END
2     ELSE
3         BEGIN
3         EVL$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
```

```
1 %SBTTL  'EVL$OPENSNK  Open a Link to a Sink Node'
1 ROUTINE EVL$OPENSNK (SNKBLK) :NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Open a logical link to the event receiver on the sink node.
1 !       If an error occurs here, close and delete the sink.  Something
1 !       is drastically wrong if there is an error here and we do not retry.
1 !       The access function for the link finishes up in EVL$OPENDONE
1 !       and further errors are handled there.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       SNKBLK          Address of the sink block
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2     BEGIN
2
2     LOCAL
2         SNK     : REF BBLOCK,                   ! Pointer to a sink block
2         STATUS                                  ! Status return
2         ;
2
2     SNK = .SNKBLK;                              ! Set local pointer
2     IF .SNK [SNK$V_STS_OPN]                     ! If its already open,
2         OR
2         .SNK [SNK$V_STS_BSY]                    ! or somethings going on
2     THEN
2         RETURN                                  ! Just ignore this call
2     ;
2
2     IF .EVL$GL_LOGMASK [ELG$V_SNKOPN]           ! Log open events?
2     THEN
3         BEGIN
3         SIGNAL                                  ! Log this open attempt
3             (
3             EVL$_LOGOPNT, 2,                    ! Message and 2 parms
3             SNK [SNK$L_SNKLEN],                 ! Ncb address
```

```
                        0                                           ! and zero for current time
                        )
                END
        ;

        STATUS = $ASSIGN                                            ! Assign a channel to net
                (
                DEVNAM = %ASCID '_NET:',
                CHAN = SNK [SNK$W_NETCHAN]                          ! Place channel in sink block
                );

        IF .STATUS                                                  ! If that worked
        THEN
                BEGIN
                SNK [SNK$V_STS_OPN] = TRUE;                         ! We have a channel tied up
                SNK [SNK$L_ROUTINE] = EVL$OPENDONE;                 ! Set the work routine
                STATUS = $QIO                                       ! and open the logical link
                        (
                        EFN = EVL$C_ASYNCH_EFN,
                        CHAN = .SNK [SNK$W_NETCHAN],                ! The channel
                        FUNC = IO$_ACCESS,                          ! access a link
                        IOSB = SNK [SNK$W_IOSB],                    ! Place status here
                        ASTADR = EVL$ASTWORK,                       ! Use the command ast routine
                        ASTPRM = .SNK,                              ! and the sink block is param
                        P2 = SNK [SNK$L_SNKLEN]                     ! The ncb is in the sink block
                        )
                END
        ;

        IF NOT EVL$NETERROR (EVL$_OPNSNK, .STATUS, 0) ! If any error
        THEN
                BEGIN
                SNK [SNK$V_STS_CLS] = TRUE;                         ! Mark for delete and
                EVL$CLOSESNK (.SNK)                                 ! Close the link
                END
        ELSE
                SNK [SNK$V_STS_BSY] = TRUE;                         ! Mark as busy

        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$:      CLRQ    -(SP)                                   ; 1712
                     0A      A2  9F 0002D          PUSHAB  10(SNK)                                 ;
                   0000'     CF  9F 00030          PUSHAB  P.AAO                                   ;
        00000000G  00        04  FB 00034          CALLS   #4, SYS$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
```

```
%SBTTL  'EVL$OPENDONE  Finish Up the Sink Open'
ROUTINE EVL$OPENDONE (SNKBLK) :NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This is the work routine which handles the completion of an
!       access function on a logical link to an event receiver on a
!       sink node.  If any error occurred, then wait for a while before
!       trying the open again.  All errors are logged by signalling them.
!       If no error occurred, then call the write event routine to
!       start writing events to the sink node if any have been queued.
!
! FORMAL PARAMETERS:
!
!       SNKBLK          Address of the sink block
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       NONE
!
!--

    BEGIN

    BIND
        OPENWAITTIME =                          ! Time to wait before trying
            UPLIT (-60*10*1000*1000, -1)        ! open again = 1 minute
        ;

    LOCAL
        SNK     : REF BBLOCK,                   ! Local pointer to sink block
        STATUS                                  ! Local status
        ;

    SNK = .SNKBLK;                              ! Set the local pointer
    IF NOT                                      ! If there was an error
        EVL$NETERROR                            ! Report it and
            (EVL$_OPNSNK, TRUE, SNK [SNK$W_IOSB] )
    THEN
        BEGIN
        IF NOT
            (STATUS = WKQ$ADD_TIMED_WORK        ! Schedule close to reopen
                (                               ! sink later
```

```
; 1819          1799 4                  EVL$CLOSESNK,                   ! Routine to activate
; 1820          1800 4                  .SNK,                           ! Its parameters
; 1821          1801 4                  0,                              ! Dummy parm
; 1822          1802 4                  OPENWAITTIME,                   ! Time to wait
; 1823          1803 4                  .SNK                            ! Timer Request id
; 1824          1804 4                  )
; 1825          1805 4              )
; 1826          1806 3          THEN
; 1827          1807 3              SIGNAL (EVL$_WKQERR, 0, .STATUS)    ! Signal something wrong
; 1828          1808 3          END
; 1829          1809 2      ELSE
; 1830          1810 3          BEGIN
; 1831          1811 3          SNK [SNK$V_STS_BSY] = FALSE;            ! Link is not busy
; 1832          1812 3          EVL$WRITEVT (.SNK)                      ! No error, so write events
; 1833          1813 3          END
; 1834          1814 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
```

```
1 %SBTTL 'EVL$CLOSESNK  Close Link to a Sink Node'
1 ROUTINE EVL$CLOSESNK (SNKBLK) :NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Close the logical link to a sink block by deassigning the channel
1 !       to net.  This causes an abort.  This is preferrable to a deaccess
1 !       function to the netacp and is certainly much simpler for us.
1 !       There are evidently theoretical problems associated with synchronous
1 !       disconnects on logical links.  The abort is guaranteed to work
1 !       correctly.  The outstanding read in the receiver gets an ABORT
1 !       status on its read and simply breaks the link.
1 !
1 !       When the link is broken, if the sink block has DELete or CLoSe status
1 !       set, deallocate the sinkblock since we are done with it.  Otherwise
1 !       call writevt to reopen the link if any events are waiting.
1 !       This mechanism allows us to temporarily close down links to sink nodes
1 !       which are very seldom used.  If a link has not been used in some time,
1 !       simply calling closesnk will close it until events are queued to it.
1 !
1 !       If the last sink block is closed, we set a flag to indicate we are not
1 !       doing any work now.  The main line may elect to exit if nobody is
1 !       doing any work.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       SNKBLK          Address of the sink block
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2    BEGIN
2
2    LOCAL
2       SNK     : REF BBLOCK,                           ! Pointer to sink block
2       STATUS                                          ! Local status return
2       ;
2
2    SNK = .SNKBLK;                                     ! Set pointer to sink
2    IF .SNK [SNK$V_STS_OPN]                            ! If sink is open
```

```
     1872  2      THEN
     1873  3          BEGIN                                           ! Deassign channel to net
P    1874  3          STATUS = $DASSGN                                ! to close logical link
P    1875  3              (
P    1876  3              CHAN = .SNK [SNK$W_NETCHAN]                 ! Channel is in snk block
     1877  3              );
     1878  3          EVL$NETERROR (EVL$_NETDAS, .STATUS, 0);        ! Report but ignore error
     1879  3          SNK [SNK$V_STS_OPN] = FALSE                     ! Link is not open
     1880  3          END
     1881  2      ;
     1882  2
     1883  2  !
     1884  2  !       The close could have been called by a timed work or for other
     1885  2  !       reasons such as the database entry for the sink went away.
     1886  2  !       If the timer bit is set, the timer may be running.  We cannot
     1887  2  !       findout if it was the timer that called us or not, so we must
     1888  2  !       try to cancel the timer entry.  If one is outstanding and we
     1889  2  !       deallocate the sink block, bad news when it goes off.
     1890  2  !
     1891  2
     1892  2      IF .SNK [SNK$V_STS_TMR]                             ! If the timer is running
     1893  2      THEN                                                ! Cancel the work item even
     1894  3          BEGIN
     1895  3          WKQ$CANCEL_TIMED_WORK (.SNK);                   ! if it is what called us.
     1896  3          SNK [SNK$V_STS_TMR] = FALSE                     ! Clear the bit
     1897  3          END
     1898  2      ;
     1899  2
     1900  2      IF  .SNK [SNK$V_STS_DEL]                            ! If link is to be deleted
     1901  2          OR
     1902  2          .SNK [SNK$V_STS_CLS]                            ! or closed
     1903  2      THEN
     1904  2          EVL$DEALLOCSNK (.SNK)                           ! Deallocate the sink block
     1905  2      ELSE
     1906  3          BEGIN
     1907  3          SNK [SNK$V_STS_BSY] = FALSE;                    ! Nothing active now
     1908  3          EVL$WRITEVT (.SNK)                              ! Otherwise check for events
     1909  3          END                                             ! and if so open link to write
     1910  2      ;                                                   ! them out
     1911  2
     1912  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
                           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
                           52  DD 00040 3$:  PUSHL   SNK                                ; 1904
              F8EB CF      01  FB 00042      CALLS   #1, EVL$DEALLOCSNK                 ;
                               04 00047      RET                                        ;
                   63      02  8A 00048 4$:  BICB2   #2, (R3)                           ; 1907
                           52  DD 0004B      PUSHL   SNK                                ; 1908
              FDD8 CF      01  FB 0004D      CALLS   #1, EVL$WRITEVT                    ;
                               04 00052      RET                                        ; 1912

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

```
1 %SBTTL  'EVL$ALLOCDBK  Allocate a Data Block'
1 GLOBAL ROUTINE EVL$ALLOCDBK (DBKSIZE, PRED, DBKRTN) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Allocate a data block with a specified size and link it into a
1 !       queue in a specified place.  If there is an error, signal it and
1 !       then exit.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       DBKSIZE         Size of data block
1 !       PRED            Predecessor in the queue
1 !       DBKRTN          Address to return address of allocated dbk
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       Routine returns true if entry is first in the queue
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN INSQUE ;
2
2     LOCAL
2         STATUS,                                   ! Hold on to status here
2         DBK : REF BBLOCK                          ! Local for the data block
2         ;
2
2     IF NOT
3         (STATUS = LIB$GET_VM (DBKSIZE, DBK) )     ! Obtain a block of data
2     THEN
3         BEGIN
3         SIGNAL (EVL$_INSFVM, 0, .STATUS);         ! Unable to obtain memory,
4         $EXIT (CODE = EVL$_INSFVM)                ! Its fatal so exit.
3         END
2     ;
2
2     CH$FILL (0, .DBKSIZE, .DBK);                  ! Zero the area
2     DBK [DBK$W_SIZE] = .DBKSIZE;                  ! Set the size in the block
2     .DBKRTN = .DBK;                               ! Return the address
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$:        MOVL     DBK, R6                                       ; 1966
    04   AC          00    6E           00   2C 00034            MOVC5    #0, (SP), #0, DBKSIZE, (R6)                   ;
                                        66      0003A                                                                   ;
                     08    A6      04   AC   B0 0003B            MOVW     DBKSIZE, 8(R6)                                ; 1967
                     0C    BC           56   D0 00040            MOVL     R6, @DBKRTN                                   ; 1968
                                        50   D4 00044            CLRL     R0                                            ; 1969
                     08    BC           66   0E 00046            INSQUE   (R6), @PRED                                   ;
                                        02   12 0004A            BNEQ     2$                                            ;
                                        50   D6 0004C            INCL     R0                                            ;
                                             04 0004E 2$:        RET                                                    ; 1971

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

```
1 %SBTTL  'EVL$DEALLOCDBK  Deallocate Data Block'
1 GLOBAL ROUTINE EVL$DEALLOCDBK (DBKADR) :NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Deallocate a data block.  This size is in the data block.
1 !       The data block is removed from the queue first.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       DBKADR          Address of the dbk
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
2     BEGIN
2
2     BUILTIN REMQUE ;
2
2     LOCAL
2         SIZE,                               ! Size of the data block
2         DBK : REF BBLOCK                    ! Local pointer to block
2         ;
2
2     REMQUE (.DBKADR, DBK);                  ! Unlink the block
2     SIZE = .DBK [DBK$W_SIZE];               ! Save its size
2     LIB$FREE_VM (SIZE, DBK)                 ! Free its memory
2
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
```

```
%SBTTL 'EVL$ASTWORK  General AST Routine to Queue Work'
ROUTINE EVL$ASTWORK (ASP) :NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine is called as an AST routine on the completion of
!       an I/O request of some kind.  The parameter is an ASP block
!       or a SNK block (these have the same header format).  This block
!       has a routine address in the header which is the address of the
!       routine to queue as work.  The ASP block is passed as a parameter
!       to the routine.  No processing or error checking is done at AST
!       level.  This synchronizes all operations and avoids any possibility
!       of race conditions with AST routines.
!
! FORMAL PARAMETERS:
!
!       ASP             Address of ASP block
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       NONE
!
!--

    BEGIN

    MAP
        ASP : REF BBLOCK
        ;

    LOCAL
        STATUS
        ;

    IF .ASP [ASP$L_ROUTINE] LSSU 512
    THEN
        SIGNAL (EVL$_WKQERR, 0, SS$_ACCVIO)     ! Some error occurred
    ;
    IF NOT
        ( STATUS = WKQ$ADD_WORK_ITEM    ! Add item to work queue
            (
            .ASP [ASP$L_ROUTINE],       ! Routine address
            .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$SIGNAL, R3                         ;
                52          04   AC   D0 00010               MOVL    AP, R2                                 ; 2065
      00000200  8F          14   A2   D1 00014               CMPL    20(R2), #512                           ;
                               09   1E 0001C               BGEQU   1$                                     ;
                               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
```

0156 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

EVLJULIAN
LIS

EVCDEF
LIS

ERRMSG
LIS

EVLIBRARY
LIS

EVLTRANS
LIS

EVLSHOW
LIS

EVLMAIN
LIS

RECEIVER
LIS