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**RECEIVER

N 13

```
RRRRRRRR    EEEEEEEEEE    CCCCCCCC  EEEEEEEEEE  IIIIII  VV        VV  EEEEEEEEEE  RRRRRRRR
RRRRRRRR    EEEEEEEEEE    CCCCCCCC  EEEEEEEEEE  IIIIII  VV        VV  EEEEEEEEEE  RRRRRRRR
RR      RR  EE          CC          EE            II    VV        VV  EE          RR      RR
RR      RR  EE          CC          EE            II    VV        VV  EE          RR      RR
RR      RR  EE          CC          EE            II    VV        VV  EE          RR      RR
RR      RR  EE          CC          EE            II    VV        VV  EE          RR      RR
RRRRRRRR    EEEEEEEE    CC          EEEEEEEE      II    VV        VV  EEEEEEEE    RRRRRRRR
RRRRRRRR    EEEEEEEE    CC          EEEEEEEE      II    VV        VV  EEEEEEEE    RRRRRRRR
RR  RR      EE          CC          EE            II    VV        VV  EE          RR  RR
RR  RR      EE          CC          EE            II    VV        VV  EE          RR  RR
RR    RR    EE          CC          EE            II       VV  VV     EE          RR    RR    ....
RR    RR    EE          CC          EE            II       VV  VV     EE          RR    RR    ....
RR      RR  EEEEEEEEEE    CCCCCCCC  EEEEEEEEEE  IIIIII       VV       EEEEEEEEEE  RR      RR  ....
RR      RR  EEEEEEEEEE    CCCCCCCC  EEEEEEEEEE  IIIIII       VV       EEEEEEEEEE  RR      RR  ....

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
```

```
MODULE receiver (IDENT = 'V04-000') =
BEGIN

!
!****************************************************************************
!*                                                                          *
!*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
!*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
!*  ALL RIGHTS RESERVED.                                                    *
!*                                                                          *
!*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
!*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
!*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
!*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
!*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
!*  TRANSFERRED.                                                            *
!*                                                                          *
!*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
!*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
!*  CORPORATION.                                                            *
!*                                                                          *
!*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
!*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************

!++
! FACILITY:  DECnet V2.0 event logger
!
! ABSTRACT:
!
!       This module contains the routines which receive events from
!       both the local node or other nodes and place the events into
!       the appropriate sink (console, file or monitor process).
!
! ENVIRONMENT:
!
!       VAX/VMS operating system. unprivileged user mode,
!
! AUTHOR:  Tim Halvorsen, June 1980
!
! Modified by:
!
!       V004    MKP0001         Kathy Perko     24-June-1984
!               Increase size of OPCOM message buffer to prevent truncation
!               of events logged to OPCOM.
!
!       V003    TMH0003         Tim Halvorsen   20-Jul-1983
!               Pass back our version number in connect accept to remote
!               event transmitter to conform to the architecture, and allow
!               PLUTOs to send us events (they didn't like our lack of
!               version number).
!               Always send events to the OPCOM facility as well as
!               sending them to the monitor process (if any); rather
!               than doing one or the other.
!
```

```
!       V002    TMH0002         Tim Halvorsen   02-May-1983
!               Fix bug which left links permanently accessed but in
!               the closed state; the links weren't being deaccessed
!               if an error was encountered issuing a link receive.
!
!       V001    TMH0001         Tim Halvorsen   03-Jun-1982
!               Convert to use new NETACP control QIO interface.
!--

!
! Include files
!

LIBRARY 'SYS$LIBRARY:STARLET';          ! VAX/VMS common definitions

REQUIRE 'SYS$LIBRARY:UTILDEF';          ! Common VMS definitions

LIBRARY 'SHRLIB$:NET';                  ! Network control QIO definitions
                                        ! (NETACP database symbols)

LIBRARY 'LIB$:EVCDEF';                  ! Event data definitions

REQUIRE 'LIB$:EVLDEF';                  ! Event logger definitions
```

```
!
! Table of contents
!

FORWARD ROUTINE
    evl$receive:          NOVALUE,     ! Receiver initialization
    update_sinkdata:      NOVALUE,     ! Update our version of sink data
    delete_sink:          NOVALUE,     ! Delete a sink
    wait_for_interrupt:   NOVALUE,     ! Wait for network interrupt
    net_interrupt:        NOVALUE,     ! Network interrupt AST routine
    open_event_link:      NOVALUE,     ! Open incoming event logical link
    close_event_link:     NOVALUE,     ! Close event logical link
    shutdown:             NOVALUE,     ! Shut down receiver gracefully
    wait_for_event:       NOVALUE,     ! Wait for an incoming event record
    event_received:       NOVALUE,     ! Accept incoming event record
    evl$queue_event:      NOVALUE,     ! Queue an event record to sinks
    output_events:        NOVALUE,     ! Output all events for a given sink
    output_console:       NOVALUE,     ! Output event to console sink
    write_line:           NOVALUE,     ! Write line to console sink device
    output_file:          NOVALUE,     ! Output event to file sink
    open_file:            NOVALUE,     ! Open file sink
    close_file:           NOVALUE,     ! Close file sink
    output_opcom:         NOVALUE,     ! Output event to OPCOM terminals
    output_monitor:       NOVALUE,     ! Output event to process sink
    save_lines:           NOVALUE,     ! Append output lines into buffer
    open_monitor:         NOVALUE,     ! Open monitor process sink
    close_monitor:        NOVALUE;     ! Close monitor process sink

!
! BUILTIN functions
!

BUILTIN
    INSQUE,                            ! INSQUE instruction
    REMQUE;                            ! REMQUE instruction

!
! Define macro for message reporting
!

MACRO
    msg(ident) =
        BEGIN
        %IF NOT %DECLARED(%NAME('evl$_',ident))
                %THEN EXTERNAL LITERAL %NAME('evl$_',ident); %FI
        %NAME('evl$_',ident)
        END%;

!
! Literals
!

LITERAL
    object_number = 26,                ! EVL object number
    max_queued_events = 10,            ! Maximum events on sink queue before
                                       ! incoming events are blocked and the
                                       ! sink queue is emptied
```

```
        mbx_maxmsg = 64;                          ! Maximum size of mailbox message

    !
    ! OWN storage
    !

    OWN
        net_channel:        WORD,                 ! Logical link channel number
        mbx_channel:        WORD,                 ! Mailbox channel number
        mbx_message:        VECTOR [mbx_maxmsg,BYTE], ! Mailbox input buffer
        iosb:               BBLOCK [8],           ! I/O status block for MBX
        sink_header:        VECTOR [2],           ! Listhead for sink descriptors
        iec_header:         VECTOR [2];           ! Listhead for incoming links

    GLOBAL
        evl$b_rcvdone:      BYTE INITIAL(true);   ! True if receiver inactive

    !
    ! External storage
    !

    EXTERNAL
        evl$gl_logmask:     BBLOCK,               ! Logging bit mask
        evl$gt_localnode;                         ! Local node name string

    !
    ! External routines
    !

    EXTERNAL ROUTINE
        format_event:       NOVALUE,              ! Format an event record
        evl$allocdbk,                             ! Allocate data storage block
        evl$deallocdbk,                           ! Deallocate data storage block
        evl$unjulian,                             ! Convert julian half-day time
        evl$printlog,                             ! Print hex bytes to log
        evl$netshow,                              ! Get information from NETACP
        wkq$add_work_item,                        ! Add work to work queue
        lib$asn_wth_mbx: ADDRESSING_MODE(GENERAL),    ! Assign with assoc. mailbox
        lib$put_output: ADDRESSING_MODE(GENERAL),     ! Write to SYS$OUTPUT
        lib$get_vm: ADDRESSING_MODE(GENERAL),         ! Allocate storage
        lib$free_vm: ADDRESSING_MODE(GENERAL);        ! Deallocate storage
```

```
  1 GLOBAL ROUTINE evl$receive: NOVALUE =
  1
  1 !---
  1 !
  1 !       This routine is called to initialize the event
  1 !       receiver and setup work to do asynchronously
  1 !       via the work queue.  A request for sink data is
  1 !       issued to NETACP and nothing is done until the
  1 !       request is satisfied.  When sink data is obtained
  1 !       (or when there is any), then the sink list is
  1 !       updated and processing of events begins.
  1 !
  1 ! Inputs:
  1 !
  1 !       None
  1 !
  1 ! Outputs:
  1 !
  1 !       None
  1 !---
  1
  2 BEGIN
  2
  2 LOCAL
  2     nfb:        BBLOCK [5],                 ! Network function block for DECLOBJ
  2     nfb_desc:   VECTOR [2],                 ! Descriptor of NFB
  2     mbxmodes:   VECTOR [2],                 ! Enables mailbox message types
  2     status;
  2
  2 sink_header [0] = sink_header;              ! Initialize sink listhead
  2 sink_header [1] = sink_header;
  2
  2 iec_header [0] = iec_header;                ! Initialize incoming channel listhead
  2 iec_header [1] = iec_header;
  2
  2 status = LIB$ASN_WTH_MBX(%ASCID '_NET:',   ! Assign channel to NETACP
  2                     0,0,                    ! mailbox MAXMSG,BUFQUO (ignored)
  2                     net_channel,            ! Channel to NETACP
  2                     mbx_channel);           ! Channel to mailbox
  2
  2 IF NOT .status                              ! If error assigning channel,
  2 THEN
  3     BEGIN
  3     SIGNAL(msg(netasn), 0, .status);        ! then signal the error
  3     RETURN;
  3     END;
  2
  2 nfb [0,0,8,0] = NFB$C_DECLOBJ;              ! Set function to "Declare object"
  2 nfb [1,0,32,0] = object_number;             ! Object number for event logger
  2
  2 nfb_desc [0] = 5;                           ! Setup descriptor of NFB
  2 nfb_desc [1] = nfb;
  2
P 2 status = $QIOW(FUNC = IO$_ACPCONTROL,       ! Issue read on mailbox
P 2             CHAN = .net_channel,
P 2             EFN = evl$c_synch_efn,
P 2             IOSB = iosb,
```

```
;     2           P1 = nfb_desc);
;     2
;     2   IF .status                              ! If successfully submitted,
;     2   THEN
;     2       status = .iosb [0,0,16,0];          ! then pick up final status
;     2
;     2   IF .status EQL ss$_badparam             ! If object already defined
;     2                                           ! (EVL is already running somewhere)
;     2   THEN
;     2       $EXIT(CODE = ss$_normal);           ! then exit process quietly
;     2
;     2   IF NOT .status
;     2   THEN
;     2       SIGNAL(msg(netasn), 0, .status);    ! then signal error
;     2
;     2   mbxmodes [0] = 0;                       ! Enable all mailbox messages
;     2   mbxmodes [1] = -1;
;     2
;   P 2   status = $QIOW(FUNC = IO$_SETMODE,      ! Issue read on mailbox
;   P 2                 CHAN = .net_channel,
;   P 2                 EFN = evl$c_synch_efn,
;   P 2                 IOSB = iosb,
;     2                 P1 = mbxmodes);
;     2
;     2   IF .status                              ! If successfully submitted,
;     2   THEN
;     2       status = .iosb [0,0,16,0];          ! then pick up final status
;     2
;     2   IF NOT .status
;     2   THEN
;     2       SIGNAL(msg(netasn), 0, .status);    ! then signal error
;     2
;     2   update_sinkdata();                      ! Update sink information
;     2
;     2   wait_for_interrupt();                   ! Wait for connect request
;     2
;     1   END;


                                                      .TITLE  RECEIVER
                                                      .IDENT  \V04-000\

                                                      .PSECT  $PLIT$,NOWRT,NOEXE,2

 00  00  00  3A  54  45  4E  5F  00000 P.AAB:        .ASCII  \_NET:\<0><0><0>          ;
                      010E0005  00008 P.AAA:        .LONG   17694725                   ;
                      00000000' 0000C               .ADDRESS P.AAB                     ;

                                                      .PSECT  $OWN$,NOEXE,2

                                 00000 NET_CHANNEL:
                                                      .BLKB   2
                                 00002 MBX_CHANNEL:
                                                      .BLKB   2
                                 00004 MBX_MESSAGE:
                                                      .BLKB   64
                                 00044 IOSB:          .BLKB   8
```

```
                                       0004C SINK_HEADER:
                                                     .BLKB     8
                                       00054 IEC_HEADER:
                                                     .BLKB     8

                                                     .PSECT    $GLOBAL$,NOEXE,2

                                  01   00000 EVL$B_RCVDONE::
                                                     .BYTE     1                                          ;

                                                     .EXTRN    EVL$GL_LOGMASK, EVL$GT_LOCALNODE
                                                     .EXTRN    FORMAT_EVENT, EVL$ALLOCDBK
                                                     .EXTRN    EVL$DEALLOCDBK, EVL$UNJULIAN
                                                     .EXTRN    EVL$PRINTLOG, EVL$NETSHOW
                                                     .EXTRN    WKQ$ADD_WORK_ITEM
                                                     .EXTRN    LIB$ASN_WTH_MBX
                                                     .EXTRN    LIB$PUT_OUTPUT, LIB$GET_VM
                                                     .EXTRN    LIB$FREE_VM, EVL$_NETASN
                                                     .EXTRN    SYS$QIOW, SYS$EXIT

                                                     .PSECT    $CODE$,NOWRT,2

                              007C     00000         .ENTRY    EVL$RECEIVE, Save R2,R3,R4,R5,R6        ; 0762
          56 00000000G   00     9E     00002         MOVAB     SYS$QIOW, R6                             ;
          55 00000000G   00     9E     00009         MOVAB     LIB$SIGNAL, R5                           ;
          54 00000000G   8F     D0     00010         MOVL      #EVL$_NETASN, R4                         ;
          53      0000'  CF     9E     00017         MOVAB     IOSB, R3                                 ;
          5E             18     C2     0001C         SUBL2     #24, SP                                  ;
     08   A3        08   A3     9E     0001F         MOVAB     SINK_HEADER, SINK_HEADER                 ; 0791
     0C   A3        08   A3     9E     00024         MOVAB     SINK_HEADER, SINK_HEADER+4               ; 0792
     10   A3        10   A3     9E     00029         MOVAB     IEC_HEADER, IEC_HEADER                   ; 0794
     14   A3        10   A3     9E     0002E         MOVAB     IEC_HEADER, IEC_HEADER+4                 ; 0795
                    BE   A3     9F     00033         PUSHAB    MBX_CHANNEL                              ; 0797
                    BC   A3     9F     00036         PUSHAB    NET_CHANNEL                              ;
                         7E     7C     00039         CLRQ      -(SP)                                    ;
                  0000'  CF     9F     0003B         PUSHAB    P.AAA                                    ;
 00000000G  00           05     FB     0003F         CALLS     #5, LIB$ASN_WTH_MBX                      ;
            52           50     D0     00046         MOVL      R0, STATUS                               ;
            0A           52     E8     00049         BLBS      STATUS, 1$                               ; 0802
                         52     DD     0004C         PUSHL     STATUS                                   ; 0805
                         7E     D4     0004E         CLRL      -(SP)                                    ;
                         54     DD     00050         PUSHL     R4                                       ;
            65           03     FB     00052         CALLS     #3, LIB$SIGNAL                           ;
                         04     00055                RET                                                ; 0804
     10   AE             16     90     00056 1$:     MOVB      #22, NFB                                 ; 0809
     11   AE             1A     D0     0005A         MOVL      #26, NFB+1                               ; 0810
     08   AE             05     D0     0005E         MOVL      #5, NFB_DESC                             ; 0812
     0C   AE        10   AE     9E     00062         MOVAB     NFB, NFB_DESC+4                          ; 0813
                         7E     7C     00067         CLRQ      -(SP)                                    ; 0819
                         7E     7C     00069         CLRQ      -(SP)                                    ;
                         7E     D4     0006B         CLRL      -(SP)                                    ;
                    1C   AE     9F     0006D         PUSHAB    NFB_DESC                                 ;
                         7E     7C     00070         CLRQ      -(SP)                                    ;
                         53     DD     00072         PUSHL     R3                                       ;
                         38     DD     00074         PUSHL     #56                                      ;
          7E        BC   A3     3C     00076         MOVZWL    NET_CHANNEL, -(SP)                       ;
                         01     DD     0007A         PUSHL     #1                                       ;
```

```
                66        0C  FB 0007C        CALLS   #12, SYS$QIOW                 ;
                52        50  D0 0007F        MOVL    R0, STATUS                    ;
                03        52  E9 00082        BLBC    STATUS, 2$                    ; 0821
                52        63  3C 00085        MOVZWL  IOSB, STATUS                  ; 0823
                14        52  D1 00088 2$:    CMPL    STATUS, #20                   ; 0825
                          09  12 0008B        BNEQ    3$                            ;
                          01  DD 0008D        PUSHL   #1                            ; 0828
      00000000G 00        01  FB 0008F        CALLS   #1, SYS$EXIT                  ;
                09        52  E8 00096 3$:    BLBS    STATUS, 4$                    ; 0830
                          52  DD 00099        PUSHL   STATUS                        ; 0832
                          7E  D4 0009B        CLRL    -(SP)                         ;
                          54  DD 0009D        PUSHL   R4                            ;
                65        03  FB 0009F        CALLS   #3, LIB$SIGNAL                ;
                          6E  D4 000A2 4$:    CLRL    MBXMODES                      ; 0834
             04 AE        01  CE 000A4        MNEGL   #1, MBXMODES+4                ; 0835
                          7E  7C 000A8        CLRQ    -(SP)                         ; 0841
                          7E  7C 000AA        CLRQ    -(SP)                         ;
                          7E  D4 000AC        CLRL    -(SP)                         ;
                     14   AE  9F 000AE        PUSHAB  MBXMODES                      ;
                          7E  7C 000B1        CLRQ    -(SP)                         ;
                          53  DD 000B3        PUSHL   R3                            ;
                          23  DD 000B5        PUSHL   #35                           ;
                7E   BC   A3  3C 000B7        MOVZWL  NET_CHANNEL, -(SP)            ;
                          01  DD 000BB        PUSHL   #1                            ;
                66        0C  FB 000BD        CALLS   #12, SYS$QIOW                 ;
                52        50  D0 000C0        MOVL    R0, STATUS                    ;
                06        52  E9 000C3        BLBC    STATUS, 5$                    ; 0843
                52        63  3C 000C6        MOVZWL  IOSB, STATUS                  ; 0845
                09        52  E8 000C9        BLBS    STATUS, 6$                    ; 0847
                          52  DD 000CC 5$:    PUSHL   STATUS                        ; 0849
                          7E  D4 000CE        CLRL    -(SP)                         ;
                          54  DD 000D0        PUSHL   R4                            ;
                65        03  FB 000D2        CALLS   #3, LIB$SIGNAL                ;
          0000V CF        00  FB 000D5 6$:    CALLS   #0, UPDATE_SINKDATA           ; 0851
          0000V CF        00  FB 000DA        CALLS   #0, WAIT_FOR_INTERRUPT        ; 0853
                              04 000DF        RET                                   ; 0855

; Routine Size:  224 bytes,    Routine Base:  $CODE$ + 0000
```

```
ROUTINE update_sinkdata: NOVALUE =

!---
!
!       This routine is called to request an updated description
!       of the sink types from NETACP.
!
!       The format of the sink information in NETACP is:
!
!               0) Sink type (console, file or monitor process)
!               4) State (on, off or hold)
!               8) Sink name string (counted string)
!
! Inputs:
!
!       None
!
! Outputs:
!
!       sink_header = Address of sink listhead
!       evl$b_rcvdone = True if list empty (receiver inactive), else false
!
!---

BEGIN

LOCAL
    status,
    buffer:     BBLOCK [256],               ! Buffer for sink information
    bufdesc:    VECTOR [2],                 ! Descriptor of above buffer
    position:   VECTOR [nfb$c_ctx_size,BYTE], ! Buffer for current position
    sink:       REF BBLOCK;                 ! Address of current sink entry

sink = .sink_header;                        ! Start at first sink block

WHILE .sink NEQ sink_header                 ! Until end of linked list,
DO
    BEGIN
    sink [sink$v_delete] = true;            ! Mark for possible deletion
    sink = .sink [sink$l_link];             ! and link to next one
    END;

bufdesc [0] = 256;                          ! Setup descriptor
bufdesc [1] = buffer;

position <0,16> = 0;                        ! Start at first ESI record

WHILE evl$netshow(
        nfb$c_db_esi,                       ! Event sink information
        nfb$c_wildcard,0,                   ! Search all records
        position,                           ! Get next record; update position
        3,                                  ! # fields for each item; fields are:
        UPLIT(  nfb$c_esi_snk,              ! Sink type (longword)
                nfb$c_esi_sta,              ! Sink state (longword)
                nfb$c_esi_lna),             ! Sink name (string)
        bufdesc)                            ! Return buffer descriptor
DO
```

```
3   BEGIN
3   IF .evl$gl_logmask [elg$v_dbupdat]  ! If we are logging database updates,
3   THEN
4       BEGIN
4       SIGNAL(msg(logdbur), 1, 0);     ! then signal update occurred
4       SIGNAL(msg(dbcrcv), 4,
4               CASE .buffer [0,0,32,0] FROM sink$c_console TO sink$c_monitor
4               OF
4                   SET
4                   [sink$c_console]:   UPLIT BYTE(%ASCIC 'console');
4                   [sink$c_file]:      UPLIT BYTE(%ASCIC 'file');
4                   [sink$c_monitor]:   UPLIT BYTE(%ASCIC 'monitor process');
4                   [OUTRANGE]:         UPLIT BYTE(%ASCIC 'unknown');
4                   TES,
4               .buffer [8,0,16,0], buffer [10,0,0,0],
4               CASE .buffer [4,0,32,0] FROM sink$c_on TO sink$c_hold
4               OF
4                   SET
4                   [sink$c_on]:        UPLIT BYTE(%ASCIC 'ON');
4                   [sink$c_off]:       UPLIT BYTE(%ASCIC 'OFF');
4                   [sink$c_hold]:      UPLIT BYTE(%ASCIC 'HOLD');
4                   [OUTRANGE]:         UPLIT BYTE(%ASCIC 'UNKNOWN');
4                   TES);
3       END;
3
3   sink = .sink_header;                ! Start at first entry in list
3
3   IF
3       WHILE .sink NEQ sink_header     ! Until end of list
3       DO
4           BEGIN
4           IF .sink [sink$b_type] EQL .buffer [0,0,32,0] ! If sink type matches
4           THEN
5               BEGIN
5               LOCAL previous_state;
5               previous_state = .sink [sink$b_state];  ! Save previous state
5               sink [sink$b_state] = .buffer [4,0,32,0]; ! Store new state
5               sink [sink$b_namelen] = .buffer [8,0,16,0]; ! Store new name
5               CH$MOVE(.sink [sink$b_namelen], buffer [10,0,0,0], sink [sink$t_name]);
5               sink [sink$v_error] = false;    ! Try operations on sink again
5               IF .sink [sink$b_state] NEQ sink$c_off  ! If sink being turned on,
5                   OR .previous_state EQL sink$c_off   ! or if sink in the process
5                                                       ! of being deleted,
5               THEN
5                   sink [sink$v_delete] = false; ! Mark this sink be retained
5               EXITLOOP false;
4               END;
4           sink = .sink [sink$l_link];         ! Skip to next in chain
4           END
4                                               ! If not found in list,
3       AND .buffer [4,0,32,0] NEQ sink$c_off   ! and not in 'off' state,
3   THEN
4       BEGIN                                   ! Then add the sink
4       LOCAL
4           length;
4       length = sink$c_length;                 ! Length of new sink entry
4       signal_if_error(LIB$GET_VM(length,sink));       ! Allocate new entry
```

```
4             CH$FILL(0,sink$c_length,.sink);               ! Zero the entry
4             sink [sink$b_type] = .buffer [0,0,32,0];      ! Store sink type
4             sink [sink$b_state] = .buffer [4,0,32,0];     ! Store new state
4             sink [sink$b_namelen] = .buffer [8,0,16,0];   ! Store new name
4             CH$MOVE(.sink [sink$b_namelen], buffer [10,0,0,0], sink [sink$t_name]);
4             sink [sink$l_evtfl] = sink [sink$l_evtfl]; ! Init event queue listhead
4             sink [sink$l_evtbl] = sink [sink$l_evtfl];
4             INSQUE(.sink, .sink_header [1]);              ! Insert into queue
4             evl$b_rcvdone = false;                        ! Mark receiver active
3             END;
3
3         IF .sink [sink$b_state] EQL sink$c_on        ! If sink just turned on,
3         THEN
3             wkq$add_work_item(output_events,.sink); ! then flush any "held" events
2         END;
2
2     !
2     !       If any sinks were turned off, we flush any events queued
2     !       for that sink and then delete the sink.  These two functions
2     !       are queued on the work queue to ensure that any asynchronous
2     !       activity on that sink completes before the delete is done.
2     !
2
2     sink = .sink_header;                          ! Start at first sink again
2
2     WHILE .sink NEQ sink_header                   ! For each sink block
2     DO
3         BEGIN
3         IF .sink [sink$v_delete]                  ! If sink was turned off or omitted,
3         THEN
4             BEGIN
4             sink [sink$b_state] = sink$c_off;         ! Set state off if sink omitted
4             wkq$add_work_item(output_events,.sink); ! then flush the sink events
4             wkq$add_work_item(delete_sink,.sink);   ! and delete the sink
4             END;
3         sink = .sink [sink$l_link];               ! Skip to next in list
2         END;
2
1     END;
```

```
                                                                  .PSECT  $PLIT$,NOWRT,NOEXE,2

                      07020041  07010011  07010010  00010 P.AAC:  .LONG   117506064, 117506065, 117571649
                   65 6C 6F  73 6E 6F 63 07  0001C P.AAD:  .ASCII  <7>\console\
                             65 6C 69 66 04  00024 P.AAE:  .ASCII  <4>\file\
73 65 63 6F 72 70 20 72 6F 74  69 6E 6F 6D 0F  00029 P.AAF:  .ASCII  <15>\monitor process\
                                          73  00038
                   6E 77 6F  6E 6B 6E 75 07  00039 P.AAG:  .ASCII  <7>\unknown\
                                   4E 4F 02  00041 P.AAH:  .ASCII  <2>\ON\
                                46 46 4F 03  00044 P.AAI:  .ASCII  <3>\OFF\
                             44 4C 4F 48 04  00048 P.AAJ:  .ASCII  <4>\HOLD\
                   4E 57 4F  4E 4B 4E 55 07  0004D P.AAK:  .ASCII  <7>\UNKNOWN\

                                                                  .EXTRN  EVL$_LOGDBUR, EVL$_DBCRCV
```

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

                                  0FFC 00000 UPDATE_SINKDATA:
                                                  .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11     ; 0856
                 5B     0000G  CF  9E 00002      MOVAB   WKQ$ADD_WORK_ITEM, R11                   ;
                 5A 00000000G  00  9E 00007      MOVAB   LIB$SIGNAL, R10                          ;
                 59     0000'  CF  9E 0000E      MOVAB   SINK_HEADER, R9                          ;
                 58     0000'  CF  9E 00013      MOVAB   P.AAC, R8                                ;
                 5E      FEB4  CE  9E 00018      MOVAB   -332(SP), SP                             ;
                               69  DD 0001D      PUSHL   SINK_HEADER                              ; 0889
                 50            6E  D0 0001F 1$:  MOVL    SINK, R0                                 ; 0891
                 51            69  9E 00022      MOVAB   SINK_HEADER, R1                          ;
                 51            50  D1 00025      CMPL    R0, R1                                   ;
                               09  13 00028      BEQL    2$                                       ;
          14     A0            01  88 0002A      BISB2   #1, 20(R0)                               ; 0894
                 6E            60  D0 0002E      MOVL    (R0), SINK                               ; 0895
                               EC  11 00031      BRB     1$                                       ; 0891
          48     AE      0100  8F  3C 00033 2$:  MOVZWL  #256, BUFDESC                            ; 0898
          4C     AE        50  AE  9E 00039      MOVAB   BUFFER, BUFDESC+4                        ; 0899
                           08  AE  B4 0003E      CLRW    POSITION                                 ; 0901
                           48  AE  9F 00041 3$:  PUSHAB  BUFDESC                                  ; 0903
                               58  DD 00044      PUSHL   R8                                       ; 0908
                               03  DD 00046      PUSHL   #3                                       ; 0903
                      14       AE  9F 00048      PUSHAB  POSITION                                 ;
                 7E            01  7D 0004B      MOVQ    #1, -(SP)                                ;
                               07  DD 0004E      PUSHL   #7                                       ;
       0000G     CF            07  FB 00050      CALLS   #7, EVL$NETSHOW                          ;
                 03            50  E8 00055      BLBS    R0, 4$                                   ;
                          011A     31 00058      BRW     23$                                      ;
                 64     0000G  CF  E9 0005B 4$:  BLBC    EVL$GL_LOGMASK, 15$                      ; 0914
                 7E            01  7D 00060      MOVQ    #1, -(SP)                                ; 0917
                    00000000G  8F  DD 00063      PUSHL   #EVL$_LOGDBUR                            ;
                 6A            03  FB 00069      CALLS   #3, LIB$SIGNAL                           ;
      02         00        54  AE  CF 0006C      CASEL   BUFFER+4, #0, #2                         ; 0928
    0018       0012      000C         00071 5$:  .WORD   6$-5$,-                                  ;
                                                         7$-5$,-                                  ;
                                                         8$-5$                                    ;
                 50        3D  A8  9E 00077      MOVAB   P.AAK, R0                                ; 0934
                               10  11 0007B      BRB     9$                                       ;
                 50        31  A8  9E 0007D 6$:  MOVAB   P.AAH, R0                                ; 0931
                               0A  11 00081      BRB     9$                                       ;
                 50        34  A8  9E 00083 7$:  MOVAB   P.AAI, R0                                ; 0932
                               04  11 00087      BRB     9$                                       ;
                 50        38  A8  9E 00089 8$:  MOVAB   P.AAJ, R0                                ; 0933
                               50  DD 0008D 9$:  PUSHL   R0                                       ;
                           5E  AE  9F 0008F      PUSHAB  BUFFER+10                                ; 0927
                 7E        60  AE  3C 00092      MOVZWL  BUFFER+8, -(SP)                          ;
      02         01        5C  AE  CF 00096      CASEL   BUFFER, #1, #2                           ; 0919
    0018       0012      000C         0009B 10$: .WORD   11$-10$,-                                ;
                                                         12$-10$,-                                ;
                                                         13$-10$                                  ;
                 50        29  A8  9E 000A1      MOVAB   P.AAG, R0                                ; 0925
                               10  11 000A5      BRB     14$                                      ;
                 50        0C  A8  9E 000A7 11$: MOVAB   P.AAD, R0                                ; 0922
                               0A  11 000AB      BRB     14$                                      ;
                 50        14  A8  9E 000AD 12$: MOVAB   P.AAE, R0                                ; 0923
                               04  11 000B1      BRB     14$                                      ;
```

```
                              50       19  A8  9E 000B3 13$:  MOVAB   P.AAF, R0                            ; 0924
                                           50  DD 000B7 14$:  PUSHL   R0                                   ;
                                           04  DD 000B9       PUSHL   #4                                   ; 0927
                       00000000G           8F  DD 000BB       PUSHL   #EVL$_DBCRCV                         ;
                              6A           06  FB 000C1       CALLS   #6, LIB$SIGNAL                       ;
                              6E           69  D0 000C4 15$:  MOVL    SINK_HEADER, SINK                    ; 0938
                              56           6E  D0 000C7 16$:  MOVL    SINK, R6                             ; 0941
                              50           69  9E 000CA       MOVAB   SINK_HEADER, R0                      ;
                              50           56  D1 000CD       CMPL    R6, R0                               ;
                                           3B  13 000D0       BEQL    19$                                  ;
   50   AE     08   A6        08           00  ED 000D2       CMPZV   #0, #8, 8(R6), BUFFER                ; 0944
                                           2D  12 000D9       BNEQ    18$                                  ;
                              57       09  A6  9A 000DB       MOVZBL  9(R6), PREVIOUS_STATE                ; 0948
                         09   A6       54  AE  90 000DF       MOVB    BUFFER+4, 9(R6)                      ; 0949
                         2E   A6       58  AE  90 000E4       MOVB    BUFFER+8, 46(R6)                     ; 0950
                              50       2E  A6  9A 000E9       MOVZBL  46(R6), R0                           ; 0951
               2F   A6   5A   AE           50  28 000ED       MOVC3   R0, BUFFER+10, 47(R6)                ;
                         14   A6           02  8A 000F3       BICB2   #2, 20(R6)                           ; 0952
                              01       09  A6  91 000F7       CMPB    9(R6), #1                            ; 0953
                                           05  12 000FB       BNEQ    17$                                  ;
                              01           57  D1 000FD       CMPL    PREVIOUS_STATE, #1                   ; 0954
                                           5F  12 00100       BNEQ    21$                                  ;
                         14   A6           01  8A 00102 17$:  BICB2   #1, 20(R6)                           ; 0957
                                           59  11 00106       BRB     21$                                  ; 0958
                              6E           66  D0 00108 18$:  MOVL    (R6), SINK                           ; 0960
                                           BA  11 0010B       BRB     16$                                  ; 0963
                         04   AE     012E  8F  3C 0010D 19$:  MOVZWL  #302, LENGTH                         ; 0968
                                           5E  DD 00113       PUSHL   SP                                   ; 0969
                                       08  AE  9F 00115       PUSHAB  LENGTH                               ;
                    00000000G 00           02  FB 00118       CALLS   #2, LIB$GET_VM                       ;
                              52           50  D0 0011F       MOVL    R0, STATUS                           ;
                              06           52  E8 00122       BLBS    STATUS, 20$                          ;
                                           52  DD 00125       PUSHL   STATUS                               ;
                              6A           01  FB 00127       CALLS   #1, LIB$SIGNAL                       ;
                                               04 0012A       RET                                          ;
                              56           6E  D0 0012B 20$:  MOVL    SINK, R6                             ; 0970
 012E   8F          00        6E           00  2C 0012E       MOVC5   #0, (SP), #0, #302, (R6)             ;
                                           66     00135                                                    ;
                         08   A6       50  AE  90 00136       MOVB    BUFFER, 8(R6)                        ; 0971
                         09   A6       54  AE  90 0013B       MOVB    BUFFER+4, 9(R6)                      ; 0972
                         2E   A6       58  AE  90 00140       MOVB    BUFFER+8, 46(R6)                     ; 0973
                              50       2E  A6  9A 00145       MOVZBL  46(R6), R0                           ; 0974
               2F   A6   5A   AE           50  28 00149       MOVC3   R0, BUFFER+10, 47(R6)                ;
                         0C   A6       0C  A6  9E 0014F       MOVAB   12(R6), 12(R6)                       ; 0975
                         10   A6       0C  A6  9E 00154       MOVAB   12(R6), 16(R6)                       ; 0976
                         04   B9           66  0E 00159       INSQUE  (R6), @SINK_HEADER+4                 ; 0977
                                    0000'  CF  94 0015D       CLRB    EVL$B_RCVDONE                        ; 0978
                              50           6E  D0 00161 21$:  MOVL    SINK, R0                             ; 0981
                                       09  A0  95 00164       TSTB    9(R0)                                ;
                                           09  12 00167       BNEQ    22$                                  ;
                                           50  DD 00169       PUSHL   R0                                   ; 0983
                                    0000V  CF  9F 0016B       PUSHAB  OUTPUT_EVENTS                        ;
                              6B           02  FB 0016F       CALLS   #2, WKQ$ADD_WORK_ITEM                ;
                                     FECC  31 00172 22$:      BRW     3$                                   ; 0903
                              6E           69  D0 00175 23$:  MOVL    SINK_HEADER, SINK                    ; 0993
                              52           6E  D0 00178 24$:  MOVL    SINK, R2                             ; 0995
                              50           69  9E 0017B       MOVAB   SINK_HEADER, R0                      ;
```

```
            50         52  D1 0017E        CMPL     R2, R0                       ;
                       1F  13 00181        BEQL     26$                          ;
            16     14  A2  E9 00183        BLBC     20(R2), 25$                  ; 0998
        09  A2         01  90 00187        MOVB     #1, 9(R2)                    ; 1001
                       52  DD 0018B        PUSHL    R2                           ; 1002
                0000V  CF  9F 0018D        PUSHAB   OUTPUT_EVENTS                ;
            6B         02  FB 00191        CALLS    #2, WK@$ADD_WORK_ITEM        ;
                       52  DD 00194        PUSHL    R2                           ; 1003
                0000V  CF  9F 00196        PUSHAB   DELETE_SINK                  ;
            6B         02  FB 0019A        CALLS    #2, WK@$ADD_WORK_ITEM        ;
            6E         62  D0 0019D 25$:   MOVL     (R2), SINK                   ; 1005
                       D6  11 001A0        BRB      24$                          ; 0995
                           04 001A2 26$:   RET                                   ; 1008

; Routine Size:  419 bytes,     Routine Base:  $CODE$ + 00E0
```

```
1009  1 ROUTINE delete_sink (sink): NOVALUE =
1010  1
1011  1 !---
1012  1 !
1013  1 !       Delete a given sink
1014  1 !
1015  1 ! Inputs:
1016  1 !
1017  1 !       sink = Address of sink block
1018  1 !
1019  1 ! Outputs:
1020  1 !
1021  1 !       None
1022  1 !---
1023  1
1024  2 BEGIN
1025  2
1026  2 MAP
1027  2     sink:       REF BBLOCK;                 ! Address of sink block
1028  2
1029  2 LOCAL
1030  2     length;
1031  2
1032  2 IF .sink [sink$l_closertn] NEQ 0            ! If there is a close routine,
1033  2 THEN
1034  2     (.sink [sink$l_closertn])(.sink);       ! Close the sink
1035  2
1036  2 REMQUE(.sink,length);                       ! Remove from linked list
1037  2
1038  2 length = sink$c_length;
1039  2 LIB$FREE_VM(length,sink);                   ! Deallocate sink block
1040  2
1041  3 evl$b_rcvdone = (.sink_header [0] EQL sink_header) ! Flag if we are active
1042  2         AND (.iec_header [0] EQL iec_header);
1043  2
1044  1 END;
```

```
                                 001C 00000 DELETE_SINK:
                                                   .WORD    Save R2,R3,R4                    ; 1009
            54      0000'  CF   9E 00002          MOVAB    SINK_HEADER, R4                  ;
            5E             04   C2 00007          SUBL2    #4, SP                           ;
            52        04   AC   D0 0000A          MOVL     SINK, R2                         ; 1032
                      22   A2   D5 0000E          TSTL     34(R2)                           ;
                           06   13 00011          BEQL     1$                               ;
                           52   DD 00013          PUSHL    R2                               ; 1034
       22   B2             01   FB 00015          CALLS    #1, @34(R2)                      ;
            6E             62   0F 00019 1$:      REMQUE   (R2), LENGTH                     ; 1036
            6E      012E   8F   3C 0001C          MOVZWL   #302, LENGTH                     ; 1038
                      04   AC   9F 00021          PUSHAB   SINK                             ; 1039
                      04   AE   9F 00024          PUSHAB   LENGTH                           ;
  00000000G 00             02   FB 00027          CALLS    #2, LIB$FREE_VM                  ;
            50             64   9E 0002E          MOVAB    SINK_HEADER, R0                  ; 1041
                           52   D4 00031          CLRL     R2                               ;
```

```
                    50        64 D1 00033       CMPL    SINK_HEADER, R0                     ;
                              02 12 00036       BNEQ    2$                                  ;
                              52 D6 00038       INCL    R2                                  ;
                    51    08  A4 9E 0003A 2$:   MOVAB   IEC_HEADER, R1                      ; 1042
                              50 D4 0003E       CLRL    R0                                  ;
                    51    08  A4 D1 00040       CMPL    IEC_HEADER, R1                      ;
                              02 12 00044       BNEQ    3$                                  ;
                              50 D6 00046       INCL    R0                                  ;
                    53        52 D2 00048 3$:   MCOML   R2, R3                              ;
        0000'  CF   50        53 8B 0004B       BICB3   R3, R0, EVL$B_RCVDONE               ;
                              04 00051          RET                                         ; 1044

; Routine Size:  82 bytes,    Routine Base:  $CODE$ + 0283
```

```
;  467        1045  1 ROUTINE wait_for_interrupt: NOVALUE =
;  468        1046  1
;  469        1047  1 !---
;  470        1048  1 !
;  471        1049  1 !       Issue an asynchronous QIO on the associated mailbox
;  472        1050  1 !       for the network channel waiting for connect requests
;  473        1051  1 !       for the receiver.
;  474        1052  1 !
;  475        1053  1 ! Inputs:
;  476        1054  1 !
;  477        1055  1 !       mbx_channel = Channel number for mailbox
;  478        1056  1 !
;  479        1057  1 ! Outputs:
;  480        1058  1 !
;  481        1059  1 !       None
;  482        1060  1 !---
;  483        1061  1
;  484        1062  2 BEGIN
;  485        1063  2
;  486        1064  2 LOCAL
;  487        1065  2     status;
;  488        1066  2
;  489      P 1067  2 status = $QIO(FUNC = IO$_READVBLK,       ! Issue read on mailbox
;  490      P 1068  2               CHAN = .mbx_channel,
;  491      P 1069  2               EFN = evl$c_asynch_efn,
;  492      P 1070  2               IOSB = iosb,
;  493      P 1071  2               ASTADR = net_interrupt,
;  494      P 1072  2               P1 = mbx_message,
;  495        1073  2               P2 = mbx_maxmsg);
;  496        1074  2
;  497        1075  2 IF NOT .status
;  498        1076  2 THEN
;  499        1077  2     SIGNAL(msg(readevt), 0, .status);   ! then signal error
;  500        1078  2
;  501        1079  1 END;


                                                .EXTRN  SYS$QIO, EVL$_READEVT

                                   0000 00000 WAIT_FOR_INTERRUPT:
                                                .WORD   Save nothing                        ; 1045
                               7E  7C 00002     CLRQ    -(SP)                               ; 1073
                               7E  7C 00004     CLRQ    -(SP)                               ;
               7E         40   8F  9A 00006     MOVZBL  #64, -(SP)                          ;
                       0000'   CF  9F 0000A     PUSHAB  MBX_MESSAGE                         ;
                               7E  D4 0000E     CLRL    -(SP)                               ;
                       0000V   CF  9F 00010     PUSHAB  NET_INTERRUPT                       ;
                       0000'   CF  9F 00014     PUSHAB  IOSB                                ;
                               31  DD 00018     PUSHL   #49                                 ;
               7E      0000'   CF  3C 0001A     MOVZWL  MBX_CHANNEL, -(SP)                  ;
                               02  DD 0001F     PUSHL   #2                                  ;
      00000000G 00             0C  FB 00021     CALLS   #12, SYS$QIO                        ;
                11             50  E8 00028     BLBS    STATUS, 1$                          ; 1075
                               50  DD 0002B     PUSHL   STATUS                              ; 1077
                               7E  D4 0002D     CLRL    -(SP)                               ;
                   00000000G   8F  DD 0002F     PUSHL   #EVL$_READEVT                       ;
```

```
          0000000G  00          03  FB 00035          CALLS   #3, LIB$SIGNAL                     ;
                                    04 0003C 1$:      RET                                        ; 1079

; Routine Size:  61 bytes,    Routine Base:  $CODE$ + 02D5
```


;

```
ROUTINE net_interrupt: NOVALUE =

!---
!
!       This AST routine is called when the outstanding QIO
!       on the associated mailbox completes.  If the interrupt
!       indicates a connect is pending, then the acceptance
!       routine is added to the work queue.
!
! Inputs:
!
!       mbx_message = Mailbox message
!
! Outputs:
!
!       None
!---

BEGIN

SELECTONEU .mbx_message [0]                     ! Select based on message type
OF
    SET

[msg$_netshut]:                                 ! Network shutting down

    BEGIN
    wkq$add_work_item(shutdown);                ! Shut down receiver gracefully
    RETURN;                                     ! Do not re-issue mailbox read
    END;

[msg$_connect]:                                 ! Incoming connect request

    BEGIN
    LOCAL
        ptr,len,
        iec:    REF BBLOCK;                     ! Incoming event channel block
    len = iec$c_length;
    LIB$GET_VM(len,iec);                        ! Allocate incoming channel block
    CH$FILL(0,iec$c_length,.iec);               ! Zero the block
    iec [iec$w_size] = iec$c_length;            ! Set length of block
    ptr = 5 + .mbx_message [4];                 ! Get index of start of ascic data
    iec [iec$b_ncblen] = .mbx_message [.ptr];   ! Set length of NCB
    CH$COPY(.mbx_message [.ptr], mbx_message [.ptr+1],
            0, iec$c_maxncblen, iec [iec$t_ncb]);
    INSQUE(.iec, .iec_header [1]);              ! Insert into list
    wkq$add_work_item(open_event_link,.iec);    ! Queue the connect accept
    evl$b_rcvdone = false;                      ! Mark receiver active
    END;

[msg$_evtrcvchg]:                               ! If event receiver database change

    wkq$add_work_item(update_sinkdata); ! then update sink information

TES;

wait_for_interrupt();                           ! Issue another read on mailbox
```

```
;  560          1137  2
;  561          1138  1 END;



                                                   01FC 00000 NET_INTERRUPT:
                                                                      .WORD   Save R2,R3,R4,R5,R6,R7,R8                 ; 1080
                            58      0000G   CF    9E 00002            MOVAB   WKQ$ADD_WORK_ITEM, R8                     ;
                            57      0000'   CF    9E 00007            MOVAB   MBX_MESSAGE, R7                           ;
                            5E              08    C2 0000C            SUBL2   #8, SP                                    ;
                            50              67    9A 0000F            MOVZBL  MBX_MESSAGE, R0                           ; 1100
                            3B              50    91 00012            CMPB    R0, #59                                   ; 1104
                                            08    12 00015            BNEQ    1$                                        ;
                                    0000V   CF    9F 00017            PUSHAB  SHUTDOWN                                  ; 1107
                            68              01    FB 0001B            CALLS   #1, WKQ$ADD_WORK_ITEM                     ;
                                                  04 0001E            RET                                               ; 1106
                            32              50    91 0001F 1$:        CMPB    R0, #50                                   ; 1111
                                            51    12 00022            BNEQ    2$                                        ;
                       04   AE      014F    8F    3C 00024            MOVZWL  #335, LEN                                 ; 1117
                                            5E    DD 0002A            PUSHL   SP                                        ; 1118
                                       08   AE    9F 0002C            PUSHAB  LEN                                       ;
                  00000000G 00              02    FB 0002F            CALLS   #2, LIB$GET_VM                            ;
                            56              6E    D0 00036            MOVL    IEC, R6                                   ; 1119
  014F    8F         00     6E              00    2C 00039            MOVC5   #0, (SP), #0, #335, (R6)                  ;
                                            66       00040                                                              ;
                       08   A6      014F    8F    B0 00041            MOVW    #335, 8(R6)                               ; 1120
                            50        04    A7    9A 00047            MOVZBL  MBX_MESSAGE+4, PTR                        ; 1121
                            50              05    C0 0004B            ADDL2   #5, PTR                                   ;
                       14   A6          6740      90 0004E            MOVB    MBX_MESSAGE[PTR], 20(R6)                  ; 1122
                            51          6740      9A 00053            MOVZBL  MBX_MESSAGE[PTR], R1                      ; 1123
  0040    8F         00     01 A740         51    2C 00057            MOVC5   R1, MBX_MESSAGE+1[PTR], #0, #64, 21(R6)   ; 1124
                                       15   A6       00060                                                              ;
                       54   B7              66    0E 00062            INSQUE  (R6), @IEC_HEADER+4                       ; 1125
                                            6E    DD 00066            PUSHL   IEC                                       ; 1126
                                    0000V   CF    9F 00068            PUSHAB  OPEN_EVENT_LINK                           ;
                            68              02    FB 0006C            CALLS   #2, @KQ$ADD_WORK_ITEM                     ;
                                    0000'   CF    94 0006F            CLRB    EVL$B_RCVDONE                             ; 1127
                                            0C    11 00073            BRB     3$                                        ; 1100
                            3F              50    91 00075 2$:        CMPB    R0, #63                                   ; 1130
                                            07    12 00078            BNEQ    3$                                        ;
                                    FD50    CF    9F 0007A            PUSHAB  UPDATE_SINKDATA                           ; 1132
                            68              01    FB 0007E            CALLS   #1, WKQ$ADD_WORK_ITEM                     ;
                     FF3D   CF              00    FB 00081 3$:        CALLS   #0, WAIT_FOR_INTERRUPT                    ; 1136
                                                  04 00086            RET                                               ; 1138

; Routine Size:  135 bytes,    Routine Base:  $CODE$ + 0312
```

```
  1 ROUTINE open_event_link (iec): NOVALUE =
  1
  1 !---
  1 !
  1 !       Open the logical link for incoming event records.
  1 !
  1 ! Inputs:
  1 !
  1 !       iec = Address of incoming event channel block
  1 !
  1 ! Outputs:
  1 !
  1 !       routine = True if link established, false if not
  1 !---
  1
  2 BEGIN
  2
  2 MAP
  2     iec:            REF BBLOCK;             ! Address of event channel block
  2
  2 LOCAL
  2     status,
  2     ptr,
  2     ncb_desc:   VECTOR [2];                 ! Descriptor of NCB
  2
  2 !
  2 !       Setup NCB for connect accept
  2 !
  2
  2 ncb_desc [0] = .iec [iec$b_ncblen];        ! Get length of requestor ncb
  2 ncb_desc [1] = iec [iec$t_ncb];            ! and address of ncb
  2
  2 !
  2 !       Get copy of NCB up to slash to enable its use in error reporting
  2 !
  2
  2 ptr = CH$FIND_CH(.iec [iec$b_ncblen], iec [iec$t_ncb], '/');
  2
  2 IF NOT CH$FAIL(.ptr)                       ! If ending slash found,
  2 THEN
  2     iec [iec$b_ncblen] = .ptr - iec [iec$t_ncb]; ! then truncate rest of junk
  2
  2 !
  2 !       Accept the connect request always since we try to support ANY version
  2 !       of remote event transmitter.  Pass back our version number in the
  2 !       "connect accept" optional data.
  2 !
  2
  2 CH$MOVE(4, UPLIT BYTE(3, 4,0,0), .ptr+3); ! Pass our version # in optional data
  2
P 2 status = $ASSIGN(DEVNAM = %ASCID '_NET:', ! Get channel for incoming link
  2                 CHAN = iec [iec$w_chan]);
  2
  2 IF NOT .status                             ! If error assigning channel,
  2 THEN
  3     BEGIN
  3     SIGNAL(msg(netasn), 0, .status);       ! then signal the error
```

```
;  620          1196 3      close_event_link(.iec);           ! and deallocate the storage
;  621          1197 3      RETURN;
;  622          1198 2      END;
;  623          1199 2
;  624        P 1200 2 status = $QIOW(FUNC = IO$_ACCESS,       ! Accept the logical link
;  625        P 1201 2                CHAN = .iec [iec$w_chan],
;  626        P 1202 2                EFN = evl$c_synch_efn,
;  627        P 1203 2                IOSB = iec [iec$w_iosb],! Address of I/O status block
;  628          1204 2                P2 = ncb_desc);         ! Address of network control block
;  629          1205 2
;  630          1206 2 IF .status                             ! If successfully submitted,
;  631          1207 2 THEN
;  632          1208 2      status = .iec [iec$w_iosb];       ! then pick up QIO final status
;  633          1209 2
;  634          1210 2 IF NOT .status                         ! If error starting up link
;  635          1211 2 THEN
;  636          1212 3      BEGIN
;  637          1213 3      SIGNAL(msg(netasn), 0, .status);  ! then signal the error
;  638          1214 3      close_event_link(.iec);           ! and deallocate the storage
;  639          1215 3      RETURN;
;  640          1216 2      END;
;  641          1217 2
;  642          1218 2 !
;  643          1219 2 !       Log the incoming connect accept
;  644          1220 2 !
;  645          1221 2
;  646          1222 2 IF .evl$gl_logmask [elg$v_rcvccf]       ! If logging receiver incoming links,
;  647          1223 2 THEN
;  648          1224 3      BEGIN
;  649          1225 3      ncb_desc [0] = .iec [iec$b_ncblen]; ! Setup descriptor of NCB - optional
;  650          1226 3      SIGNAL(msg(logopnr), 3, .ncb_desc [0], .ncb_desc [1], 0);
;  651          1227 3      END;
;  652          1228 2
;  653          1229 2 wait_for_event(.iec);                  ! Wait for an incoming event record
;  654          1230 2
;  655          1231 1 END;


                                                          .PSECT   $PLIT$,NOWRT,NOEXE,2

                         00  00  04  03  00055 P.AAL:     .BYTE    3, 4, 0, 0                        ;
                                         00059            .BLKB    3
         00  00  00  3A  54  45  4E  5F  0005C P.AAN:     .ASCII   \_NET:\<0><0><0>                  ;
                               010E0005  00064 P.AAM:     .LONG    17694725                          ;
                              00000000'  00068            .ADDRESS P.AAN                             ;

                                                          .EXTRN   SYS$ASSIGN, EVL$_LOGOPNR

                                                          .PSECT   $CODE$,NOWRT,2

                                   003C  00000 OPEN_EVENT_LINK:
                                                          .WORD    Save R2,R3,R4,R5                  ; 1139
                  55 00000000G   00  9E  00002            MOVAB    LIB$SIGNAL, R5                    ;
                  54 00000000G   8F  D0  00009            MOVL     #EVL$_NETASN, R4                  ;
                  5E             04  C2  00010            SUBL2    #4, SP                            ;
                  52        04   AC  D0  00013            MOVL     IEC, R2                           ; 1168
```

```
                          7E        14  A2  9A 00017       MOVZBL  20(R2), NCB_DESC                 ;
                   04     AE        15  A2  9E 0001B       MOVAB   21(R2), NCB_DESC+4               ; 1169
                          50        14  A2  9A 00020       MOVZBL  20(R2), R0                       ; 1175
       15   A2            50            2F  3A 00024       LOCC    #47, R0, 21(R2)                  ;
                                        02  12 00029       BNEQ    1$                               ;
                                        51  D4 0002B       CLRL    R1                               ;
                                        51  D5 0002D  1$:  TSTL    PTR                              ; 1177
                                        09  13 0002F       BEQL    2$                               ;
                          50        15  A2  9E 00031       MOVAB   21(R2), R0                       ; 1179
       14   A2            51            50  83 00035       SUBB3   R0, PTR, 20(R2)                  ;
                   03     A1     0000'  CF  D0 0003A  2$:  MOVL    P.AAL, 3(PTR)                    ; 1187
                                        7E  7C 00040       CLRQ    -(SP)                            ; 1190
                                    0A  A2  9F 00042       PUSHAB  10(R2)                           ;
                                 0000'  CF  9F 00045       PUSHAB  P.AAM                            ;
                00000000G 00            04  FB 00049       CALLS   #4, SYS$ASSIGN                   ;
                          53            50  D0 00050       MOVL    R0, STATUS                       ;
                          2A            53  E9 00053       BLBC    STATUS, 3$                       ; 1192
                                        7E  7C 00056       CLRQ    -(SP)                            ; 1204
                                        7E  7C 00058       CLRQ    -(SP)                            ;
                                    10  AE  9F 0005A       PUSHAB  NCB_DESC                         ;
                                        7E  7C 0005D       CLRQ    -(SP)                            ;
                                        7E  D4 0005F       CLRL    -(SP)                            ;
                                    0C  A2  9F 00061       PUSHAB  12(R2)                           ;
                                        32  DD 00064       PUSHL   #50                              ;
                          7E        0A  A2  3C 00066       MOVZWL  10(R2), -(SP)                    ;
                                        01  DD 0006A       PUSHL   #1                               ;
                00000000G 00            0C  FB 0006C       CALLS   #12, SYS$QIOW                    ;
                          53            50  D0 00073       MOVL    R0, STATUS                       ;
                          07            53  E9 00076       BLBC    STATUS, 3$                       ; 1206
                          53        0C  A2  3C 00079       MOVZWL  12(R2), STATUS                   ; 1208
                          11            53  E8 0007D       BLBS    STATUS, 4$                       ; 1210
                                        53  DD 00080  3$:  PUSHL   STATUS                           ; 1213
                                        7E  D4 00082       CLRL    -(SP)                            ;
                                        54  DD 00084       PUSHL   R4                               ;
                          65            03  FB 00086       CALLS   #3, LIB$SIGNAL                   ;
                                        52  DD 00089       PUSHL   R2                               ; 1214
                    0000V CF            01  FB 0008B       CALLS   #1, CLOSE_EVENT_LINK             ;
                                            04 00090       RET                                      ; 1212
            17      0000G CF            02  E1 00091  4$:  BBC     #2, EVL$GL_LOGMASK, 5$           ; 1222
                          6E        14  A2  9A 00097       MOVZBL  20(R2), NCB_DESC                 ; 1225
                                        7E  D4 0009B       CLRL    -(SP)                            ; 1226
                                    08  AE  DD 0009D       PUSHL   NCB_DESC+4                       ;
                                    08  AE  DD 000A0       PUSHL   NCB_DESC                         ;
                                        03  DD 000A3       PUSHL   #3                               ;
                            00000000G   8F  DD 000A5       PUSHL   #EVL$_LOGOPNR                    ;
                          65            05  FB 000AB       CALLS   #5, LIB$SIGNAL                   ;
                                        52  DD 000AE  5$:  PUSHL   R2                               ; 1229
                    0000V CF            01  FB 000B0       CALLS   #1, WAIT_FOR_EVENT               ;
                                            04 000B5       RET                                      ; 1231

; Routine Size:  182 bytes,    Routine Base:  $CODE$ + 0399
```

```
;  657      1232  1 ROUTINE close_event_link (iec): NOVALUE =
;  658      1233  1
;  659      1234  1 !---
;  660      1235  1 !
;  661      1236  1 !       Close the logical link for incoming event records.
;  662      1237  1 !
;  663      1238  1 ! Inputs:
;  664      1239  1 !
;  665      1240  1 !       iec = Address of incoming event channel block
;  666      1241  1 !
;  667      1242  1 ! Outputs:
;  668      1243  1 !
;  669      1244  1 !       None
;  670      1245  1 !---
;  671      1246  1
;  672      1247  2 BEGIN
;  673      1248  2
;  674      1249  2 MAP
;  675      1250  2     iec:        REF BBLOCK;                 ! Address of incoming event channel
;  676      1251  2
;  677      1252  2 LOCAL
;  678      1253  2     length,
;  679      1254  2     status;
;  680      1255  2
;  681      1256  2 IF .iec [iec$w_chan] NEQ 0               ! If channel was assigned,
;  682      1257  2 THEN
;  683      1258  3     BEGIN
;  684      1259  3     status = $DASSGN(CHAN = .iec [iec$w_chan]); ! Deassign network channel
;  685      1260  3
;  686      1261  3     IF NOT .status                              ! If error detected,
;  687      1262  3     THEN
;  688      1263  3         SIGNAL(msg(netdas), 0, .status);        ! then signal error
;  689      1264  2     END;
;  690      1265  2
;  691      1266  2 REMQUE(.iec, status);                   ! Remove from linked list
;  692      1267  2 length = .iec [iec$w_size];             ! Get size of block
;  693      1268  2 LIB$FREE_VM(length,iec);                ! Deallocate storage
;  694      1269  2
;  695      1270  3 evl$b_rcvdone = (.sink_header [0] EQL sink_header) ! Flag if we are active
;  696      1271  2         AND (.iec_header [0] EQL iec_header);
;  697      1272  2
;  698      1273  1 END;


                                                .EXTRN  SYS$DASSGN, EVL$_NETDAS

                              001C 00000 CLOSE_EVENT_LINK:
                                                .WORD   Save R2,R3,R4                          ; 1232
              54      0000'   CF   9E 00002     MOVAB   SINK_HEADER, R4                        ;
              5E              04   C2 00007     SUBL2   #4, SP                                 ;
              52        04    AC   D0 0000A     MOVL    IEC, R2                                ; 1256
                        0A    A2   B5 0000E     TSTW    10(R2)                                 ;
                              22   13 00011     BEQL    1$                                     ;
              7E        0A    A2   3C 00013     MOVZWL  10(R2), -(SP)                          ; 1259
     00000000G 00             01   FB 00017     CALLS   #1, SYS$DASSGN                         ;
              53              50   D0 0001E     MOVL    R0, STATUS                             ;
```

```
                   11           53  E8 00021        BLBS    STATUS, 1$                                 ; 1261
                                53  DD 00024        PUSHL   STATUS                                     ; 1263
                                7E  D4 00026        CLRL    -(SP)                                      ;
                      00000000G 8F  DD 00028        PUSHL   #EVL$_NETDAS                               ;
        00000000G  00           03  FB 0002E        CALLS   #3, LIB$SIGNAL                             ;
                   53           62  0F 00035  1$:   REMQUE  (R2), STATUS                               ; 1266
                   50      04   AC  D0 00038        MOVL    IEC, R0                                    ; 1267
                   6E      08   A0  3C 0003C        MOVZWL  8(R0), LENGTH                              ;
                           04   AC  9F 00040        PUSHAB  IEC                                        ; 1268
                           04   AE  9F 00043        PUSHAB  LENGTH                                     ;
        00000000G  00           02  FB 00046        CALLS   #2, LIB$FREE_VM                            ;
                   50           64  9E 0004D        MOVAB   SINK_HEADER, R0                            ; 1270
                                52  D4 00050        CLRL    R2                                         ;
                   50           64  D1 00052        CMPL    SINK_HEADER, R0                            ;
                                02  12 00055        BNEQ    2$                                         ;
                                52  D6 00057        INCL    R2                                         ;
                   51      08   A4  9E 00059  2$:   MOVAB   IEC_HEADER, R1                             ; 1271
                                50  D4 0005D        CLRL    R0                                         ;
                   51      08   A4  D1 0005F        CMPL    IEC_HEADER, R1                             ;
                                02  12 00063        BNEQ    3$                                         ;
                                50  D6 00065        INCL    R0                                         ;
                   53           52  D2 00067  3$:   MCOML   R2, R3                                     ;
      0000'   CF   50           53  8B 0006A        BICB3   R3, R0, EVL$B_RCVDONE                      ;
                                    04 00070        RET                                                ; 1273
```

; Routine Size: 113 bytes,   Routine Base: $CODE$ + 044F

```
;  700      1274  1 ROUTINE shutdown: NOVALUE =
;  701      1275  1
;  702      1276  1 !---
;  703      1277  1 !
;  704      1278  1 !       This routine is called when the network is shutting down
;  705      1279  1 !       to gracefully close all sinks and incoming links so that
;  706      1280  1 !       EVL goes away quietly.
;  707      1281  1 !
;  708      1282  1 ! Inputs:
;  709      1283  1 !
;  710      1284  1 !       None
;  711      1285  1 !
;  712      1286  1 ! Outputs:
;  713      1287  1 !
;  714      1288  1 !       None
;  715      1289  1 !---
;  716      1290  1
;  717      1291  2 BEGIN
;  718      1292  2
;  719      1293  2 LOCAL
;  720      1294  2     ptr:        REF BBLOCK,             ! Pointer to sink or iec block
;  721      1295  2     next_ptr;
;  722      1296  2
;  723      1297  2 ptr = .iec_header;                      ! Start at first link context block
;  724      1298  2 WHILE .ptr NEQ iec_header               ! Until end of linked list,
;  725      1299  2 DO
;  726      1300  3     BEGIN
;  727      1301  3     next_ptr = .ptr [iec$l_link];
;  728      1302  3     close_event_link(.ptr);             ! Abort the incoming link
;  729      1303  3     ptr = .next_ptr;                    ! and link to next in chain
;  730      1304  2     END;
;  731      1305  2
;  732      1306  2 ptr = .sink_header;                     ! Start at first sink block
;  733      1307  2 WHILE .ptr NEQ sink_header              ! Until end of linked list,
;  734      1308  2 DO
;  735      1309  3     BEGIN
;  736      1310  3     ptr [sink$b_state] = sink$c_off;            ! Force deletion of sink
;  737      1311  3     wkq$add_work_item(output_events,.ptr);      ! flush the sink events
;  738      1312  3     wkq$add_work_item(delete_sink,.ptr);        ! and delete the sink
;  739      1313  3     ptr = .ptr [sink$l_link];           ! Link to next one
;  740      1314  2     END;
;  741      1315  2
;  742      1316  1 END;



                                     001C 00000 SHUTDOWN:
                                                    .WORD    Save R2,R3,R4                    ; 1274
            54      0000'   CF  9E 00002            MOVAB    IEC_HEADER, R4                   ;
            52              64  D0 00007            MOVL     IEC_HEADER, PTR                  ; 1297
            50              64  9E 0000A 1$:        MOVAB    IEC_HEADER, R0                   ; 1298
            50              52  D1 0000D            CMPL     PTR, R0                          ;
                            0F  13 00010            BEQL     2$                               ;
            53              62  D0 00012            MOVL     (PTR), NEXT_PTR                  ; 1301
                            52  DD 00015            PUSHL    PTR                              ; 1302
```

```
        FF73  CF            01  FB 00017       CALLS   #1, CLOSE_EVENT_LINK                     ;
              52            53  D0 0001C       MOVL    NEXT_PTR, PTR                            ; 1303
                            E9  11 0001F       BRB     1$                                       ; 1298
              52      F8    A4  D0 00021 2$:   MOVL    SINK_HEADER, PTR                         ; 1306
              50      F8    A4  9E 00025 3$:   MOVAB   SINK_HEADER, R0                          ; 1307
              50            52  D1 00029       CMPL    PTR, R0                                  ;
                            1F  13 0002C       BEQL    4$                                       ;
          09  A2            01  90 0002E       MOVB    #1, 9(PTR)                               ; 1310
                            52  DD 00032       PUSHL   PTR                                      ; 1311
                    0000V   CF  9F 00034       PUSHAB  OUTPUT_EVENTS                            ;
       0000G  CF            02  FB 00038       CALLS   #2, WK0$ADD_WORK_ITEM                    ;
                            52  DD 0003D       PUSHL   PTR                                      ; 1312
                    FD80    CF  9F 0003F       PUSHAB  DELETE_SINK                              ;
       0000G  CF            02  FB 00043       CALLS   #2, WK0$ADD_WORK_ITEM                    ;
              52            62  D0 00048       MOVL    (PTR), PTR                               ; 1313
                            D8  11 0004B       BRB     3$                                       ; 1307
                                04 0004D 4$:   RET                                              ; 1316

; Routine Size:  78 bytes,    Routine Base:  $CODE$ + 04C0
```

```
;    1 ROUTINE wait_for_event (iec): NOVALUE =
;    1
;    1 !---
;    1 !
;    1 !       This routine is called to obtain processed events from
;    1 !       the receiver's incoming logical link.  Each incoming event
;    1 !       is immediately queued to the disposal queue for the appropriate
;    1 !       sink types.
;    1 !
;    1 ! Inputs:
;    1 !
;    1 !       iec = Address of incoming event channel context block
;    1 !
;    1 ! Outputs:
;    1 !
;    1 !       None
;    1 !---
;    1
;    2 BEGIN
;    2
;    2 MAP
;    2     iec:        REF BBLOCK;              ! Address of incoming event channel
;    2
;    2 LOCAL
;    2     status;
;    2
;  P 2 status = $QIO(FUNC = IO$_READVBLK,       ! Get event from incoming link
;  P 2                 CHAN = .iec [iec$w_chan],
;  P 2                 EFN = evl$c_asynch_efn,
;  P 2                 IOSB = iec [iec$w_iosb],! Address of I/O status block
;  P 2                 ASTADR = event_received,! Address of completion routine
;  P 2                 ASTPRM = .iec,          ! Giving IEC as routine parameter
;  P 2                 P1 = iec [iec$t_event], ! Address of event buffer
;    2                 P2 = iec$c_maxevtlen);  ! Length of event buffer
;    2
;    2 IF NOT .status                           ! If unsuccessful
;    2 THEN
;    3     BEGIN
;    3     SIGNAL(msg(readevt),0,.status);      ! then signal error
;    3     close_event_link(.iec);              ! close link; deallocate storage
;    2     END;
;    2
;    1 END;


                              0004 00000 WAIT_FOR_EVENT:
                                               .WORD   Save R2                      ; 1317
                        7E  7C 00002           CLRQ    -(SP)                        ; 1350
                        7E  7C 00004           CLRQ    -(SP)                        ;
        7E      FA      8F  9A 00006           MOVZBL  #250, -(SP)                  ;
        52      04      AC  D0 0000A           MOVL    IEC, R2                      ;
                55      A2  9F 0000E           PUSHAB  85(R2)                       ;
                        52  DD 00011           PUSHL   R2                           ;
             0000V      CF  9F 00013           PUSHAB  EVENT_RECEIVED               ;
```

```
                        0C   A2 9F 00017        PUSHAB  12(R2)                      ;
                             31 DD 0001A        PUSHL   #49                         ;
                7E      0A   A2 3C 0001C        MOVZWL  10(R2), -(SP)               ;
                             02 DD 00020        PUSHL   #2                          ;
       00000000G  00         0C FB 00022        CALLS   #12, SYS$QIO                ;
                  18         50 E8 00029        BLBS    STATUS, 1$                  ; 1352
                             50 DD 0002C        PUSHL   STATUS                      ; 1355
                             7E D4 0002E        CLRL    -(SP)                       ;
                  00000000G  8F DD 00030        PUSHL   #EVL$_READEVT               ;
       00000000G  00         03 FB 00036        CALLS   #3, LIB$SIGNAL              ;
                             52 DD 0003D        PUSHL   R2                          ; 1356
            FEFD  CF         01 FB 0003F        CALLS   #1, CLOSE_EVENT_LINK        ;
                                04 00044 1$:    RET                                 ; 1359

; Routine Size:  69 bytes,    Routine Base:  $CODE$ + 050E
```

```
1 ROUTINE event_received (iec): NOVALUE =
1
1 !---
1 !
1 !       This AST routine is called when a new event has come
1 !       in over the logical link.  The event is queued to the
1 !       appropriate sink type(s).
1 !
1 ! Inputs:
1 !
1 !       iec = Address of incoming event channel context block
1 !
1 ! Outputs:
1 !
1 !       None
1 !---
1
2 BEGIN
2
2 MAP
2     iec:        REF BBLOCK;              ! Address of incoming event block
2
2 IF NOT .iec [iec$w_iosb]                 ! If error from QIO,
2 THEN
3     BEGIN
3     IF .iec [iec$w_iosb] NEQ ss$_linkabort ! If link was not aborted,
3     THEN
3         SIGNAL(msg(readevt), 0, .iec [iec$w_iosb]); ! then signal the error
3     close_event_link(.iec);              ! close the link until re-established
3     RETURN;
2     END;
2
2 wkq$add_work_item(evl$queue_event,       ! Queue event to sink(s)
2     .iec [iec$w_iosb1], iec [iec$t_event]);
2
2 wkq$add_work_item(wait_for_event, .iec);! Issue another QIO only after
2                                          ! the entire queue is cleaned out
2
1 END;


                                 0004 00000 EVENT_RECEIVED:
                                                  .WORD   Save R2                          ; 1360
             52         04   AC   D0 00002        MOVL    IEC, R2                          ; 1382
             23         0C   A2   E8 00006        BLBS    12(R2), 2$
      20E4   8F         0C   A2   B1 0000A        CMPW    12(R2), #8420                    ; 1385
                             13   13 00010        BEQL    1$
             7E         0C   A2   3C 00012        MOVZWL  12(R2), -(SP)                    ; 1387
                                  7E D4 00016     CLRL    -(SP)                            ;
                 00000000G   8F   DD 00018        PUSHL   #EVL$_READEVT                    ;
 00000000G   00              03   FB 0001E        CALLS   #3, LIB$SIGNAL                   ;
                             52   DD 00025 1$:    PUSHL   R2                               ; 1388
      FED0   CF              01   FB 00027        CALLS   #1, CLOSE_EVENT_LINK
                             04      0002C        RET                                      ; 1384
```

```
                      55   A2   9F 0002D 2$:     PUSHAB  85(R2)                                      ; 1393
             7E       0E   A2   3C 00030         MOVZWL  14(R2), -(SP)                               ;
                   0000V   CF   9F 00034         PUSHAB  EVL$QUEUE_EVENT                             ; 1392
     0000G  CF             03   FB 00038         CALLS   #3, WKQ$ADD_WORK_ITEM                       ; 1393
                           52   DD 0003D         PUSHL   R2                                          ; 1395
                    FF78   CF   9F 0003F         PUSHAB  WAIT_FOR_EVENT                              ;
     0000G  CF             02   FB 00043         CALLS   #2, WKQ$ADD_WORK_ITEM                       ;
                                04 00048         RET                                                 ; 1398

; Routine Size:  73 bytes,    Routine Base:  $CODE$ + 0553
```

```
1 GLOBAL ROUTINE evl$queue_event (length, event): NOVALUE =
1
1 !---
1 !
1 !       This routine is called to queue a given event record
1 !       to the appropriate sink type(s).
1 !
1 ! Inputs:
1 !
1 !       length = Length of event record
1 !       event = Address of event record:
1 !
1 !               0) Function code (always 1 = event record)
1 !               1) Event flags (byte) which sink(s) get the event
1 !               2) Event code (word)
1 !               4) Time of event (6 bytes) in Julian half-day format
1 !               10) Source node (2 + 1-7 bytes) address and name
1 !               ?) Event entity (1+n bytes) describing line or node ID
1 !               ?) Event data, depending on event code
1 !
1 ! Outputs:
1 !
1 !       None
1 !---
1
2 BEGIN
2
2 MAP
2     event:      REF BBLOCK;               ! Address of event record
2
2 LOCAL
2     event_overflow:     BYTE,             ! True if event queue overflowed
2     sink:       REF BBLOCK;               ! Address of sink descriptor block
2
2 IF .evl$gl_logmask [elg$v_rcvevt]        ! If logging received messages,
2 THEN
3     BEGIN
3     LOCAL msgdesc: VECTOR [2];
3     msgdesc [0] = .length;                ! Setup descriptor of message
3     msgdesc [1] = .event;
3     EVL$PRINTLOG($BITPOSITION(elg$v_rcvevt),    ! Log the received message
3         %ASCID 'Event received',0,msgdesc);
2     END;
2
2 event_overflow = false;                   ! Preset no overflow
2 sink = .sink_header [0];                  ! Start at first sink entry
2
2 WHILE .sink NEQ sink_header               ! Until end of sink list
2 DO
3     BEGIN
3     IF .(event [evt$b_flags])<.sink [sink$b_type]-1,1> ! If this sink gets event,
3         AND .sink [sink$b_state] NEQ sink$c_off ! and delete not pending,
3     THEN
4         BEGIN                             ! then insert into sink's queue
4         LOCAL
4             status,
4             entry:      REF BBLOCK;       ! Address of event data block
```

```
;  885          1456  4
;  886          1457  4            status = evl$allocdbk(evq$c_size+.length, .sink [sink$l_evtbl], entry);
;  887          1458  4            sink [sink$w_events] = .sink [sink$w_events] + 1;
;  888          1459  4            entry [evq$w_evtsize] = .length;
;  889          1460  4            CH$MOVE(.length, .event, entry [evq$t_event]);
;  890          1461  4            IF .status                      ! If first entry in queue,
;  891          1462  4            THEN
;  892          1463  4                wkq$add_work_item(output_events, .sink); ! start emptying queue
;  893          1464  4            IF .sink [sink$w_events] GEQU max_queued_events ! If too many events,
;  894          1465  4            THEN
;  895          1466  4                event_overflow = true;      ! then mark events are overflowing
;  896          1467  3            END;
;  897          1468  3        sink = .sink [sink$l_link];         ! Skip to next sink type
;  898          1469  2        END;
;  899          1470  2
;  900          1471  1 END;



                                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

00 64 65 76 69 65 63 65 72 20 74 6E 65 76 45  0006C P.AAP:  .ASCII  \Event received\<0><0>                     ;
                                          00  0007B                                                            ;
                                    010E000E  0007C P.AAO:  .LONG   17694734                                   ;
                                   00000000'  00080         .ADDRESS P.AAP                                     ;



                                                                        .PSECT  $CODE$,NOWRT,2

                                 03FC 00000         .ENTRY  EVL$QUEUE_EVENT, Save R2,R3,R4,R5,R6,R7,R8,-; 1399
                                                            R9                                         ;
                     5E           0C  C2 00002      SUBL2   #12, SP                                    ;
           15     0000G  CF       06  E1 00005      BBC     #6, EVL$GL_LOGMASK, 1$                     ; 1433
                     04  AE   04  AC  7D 0000B      MOVQ    LENGTH, MSGDESC                            ; 1437
                              04  AE  9F 00010      PUSHAB  MSGDESC                                    ; 1439
                                  7E  D4 00013      CLRL    -(SP)                                      ;
                           0000'  CF  9F 00015      PUSHAB  P.AAO                                      ;
                                  06  DD 00019      PUSHL   #6                                         ;
                  0000G  CF       04  FB 0001B      CALLS   #4, EVL$PRINTLOG                           ;
                                  59  94 00020 1$:  CLRB    EVENT_OVERFLOW                             ; 1443
                     56    0000'  CF  D0 00022      MOVL    SINK_HEADER, SINK                          ; 1444
                     57       08  AC  D0 00027      MOVL    EVENT, R7                                  ; 1449
                     50    0000'  CF  9E 0002B 2$:  MOVAB   SINK_HEADER, R0                            ; 1446
                     50           56  D1 00030      CMPL    SINK, R0                                   ;
                                  51  13 00033      BEQL    5$                                         ;
                     50       08  A6  9A 00035      MOVZBL  8(SINK), R0                                ; 1449
                                  50  D7 00039      DECL    R0                                         ;
           41        01  A7       50  E1 0003B      BBC     R0, 1(R7), 4$                              ;
                     01       09  A6  91 00040      CMPB    9(SINK), #1                                ; 1450
                                  3B  13 00044      BEQL    4$                                         ;
                                  5E  DD 00046      PUSHL   SP                                         ; 1457
                              10  A6  DD 00048      PUSHL   16(SINK)                                   ;
           7E        04  AC       0C  C1 0004B      ADDL3   #12, LENGTH, -(SP)                         ;
                  0000G  CF       03  FB 00050      CALLS   #3, EVL$ALLOCDBK                           ;
                     58           50  D0 00055      MOVL    R0, STATUS                                 ;
                              0A  A6  B6 00058      INCW    10(SINK)                                   ; 1458
```

```
                          50            6E  D0 0005B         MOVL    ENTRY, R0                                  ; 1459
                     0A   A0      04    AC  B0 0005E         MOVW    LENGTH, 10(R0)                             ;
          0C   A0    08   BC      04    AC  28 00063         MOVC3   LENGTH, @EVENT, 12(R0)                     ; 1460
                          0B            58  E9 0006A         BLBC    STATUS, 3$                                 ; 1461
                                        56  DD 0006D         PUSHL   SINK                                       ; 1463
                              0000V     CF  9F 0006F         PUSHAB  OUTPUT_EVENTS                              ;
                  0000G   CF            02  FB 00073         CALLS   #2, WK0$ADD_WORK_ITEM                      ;
                          0A      0A    A6  B1 00078 3$:     CMPW    10(SINK), #10                              ; 1464
                                        03  1F 0007C         BLSSU   4$                                         ;
                          59            01  90 0007E         MOVB    #1, EVENT_OVERFLOW                         ; 1466
                          56            66  D0 00081 4$:     MOVL    (SINK), SINK                               ; 1468
                                        A5  11 00084         BRB     2$                                         ; 1446
                                            04 00086 5$:     RET                                                ; 1471
```

; Routine Size: 135 bytes,   Routine Base: $CODE$ + 059C

```
1 ROUTINE output_events (sink): NOVALUE =
1
1 !---
1 !
1 !       This routine is called when an event is queued for
1 !       any sink.  All events for the given sink are written
1 !       to the appropriate destination depending on the sink
1 !       type.  Note that this routine may be called when
1 !       there are no events queued if the sink is turned off
1 !       and wants to be cleaned up here.
1 !
1 ! Inputs:
1 !
1 !       sink = Address of sink descriptor block
1 !
1 ! Outputs:
1 !
1 !       None
1 !---
1
2 BEGIN
2
2 MAP
2     sink:       REF BBLOCK;                 ! Address of sink block
2
2 LOCAL
2     event_data: REF BBLOCK,                 ! Address of next event data block
2     event:      REF BBLOCK;                 ! Address of actual event record
2
2 IF .sink [sink$b_state] EQL sink$c_hold ! If holding sink output,
2 THEN
2     RETURN;                                 ! return without dequeueing anything
2
2 WHILE NOT REMQUE(.sink [sink$l_evtfl], event_data) ! Dequeue the next event
2 DO
3     BEGIN
3     sink [sink$w_events] = .sink [sink$w_events] - 1;    ! Decrement events left
3     event = event_data [evq$t_event];   ! Address the actual event record
3
3     IF .sink [sink$b_state] EQL sink$c_on        ! If sink is ON,
3     THEN
3     CASE .sink [sink$b_type] FROM 0 TO sink$c_monitor   ! Dispatch to routine
3     OF
3         SET
3         [sink$c_console]: output_console(        ! Output to console
3                     .sink, .event, .event_data [evq$w_evtsize]);
3         [sink$c_file]:    output_file(           ! Output to file
3                     .sink, .event, .event_data [evq$w_evtsize]);
3         [sink$c_monitor]:
4                 BEGIN
4                 output_opcom(                    ! Output to OPCOM terminals
4                     .sink, .event, .event_data [evq$w_evtsize]);
4                 output_monitor(                  ! Output to process (if any)
4                     .sink, .event, .event_data [evq$w_evtsize]);
3                 END;
3         [INRANGE]:        SIGNAL(msg(badtype), 1, .sink [sink$b_type]);
3         TES;
```

```
;  959      1529 3
;  960      1530 3         evl$deallocdbk(.event_data);          ! Deallocate data block
;  961      1531 2         END;
;  962      1532 2
;  963      1533 2  IF .sink [sink$b_type] EQL sink$c_console ! If console sink,
;  964      1534 2      AND .sink [sink$l_closertn] NEQ 0     ! and currently open,
;  965      1535 2  THEN
;  966      1536 2      (.sink [sink$l_closertn])(.sink);     ! then close the sink at the end
;  967      1537 2                                            ! of the "batch" to prevent the
;  968      1538 2                                            ! file/device from being tied up
;  969      1539 1  END;


                                                         .EXTRN  EVL$_BADTYPE

                                  001C 00000 OUTPUT_EVENTS:
                                                         .WORD   Save R2,R3,R4                       ; 1472
                      50      04   AC  D0 00002          MOVL    SINK, R0                            ; 1501
                      02      09   A0  91 00006          CMPB    9(R0), #2                           ;
                                       01  12 0000A      BNEQ    1$                                  ;
                                           04 0000C      RET                                         ;
                      54      04   AC  D0 0000D 1$:      MOVL    SINK, R4                            ; 1505
                      52      0C   B4  0F 00011 2$:      REMQUE  @12(R4), EVENT_DATA                 ;
                                       73  1D 00015      BVS     9$                                  ;
                      54      04   AC  D0 00017          MOVL    SINK, R4                            ; 1508
                              0A   A4  B7 0001B          DECW    10(R4)                              ;
                      53      0C   A2  9E 0001E          MOVAB   12(R2), EVENT                       ; 1509
                              09   A4  95 00022          TSTB    9(R4)                               ; 1511
                                       5A  12 00025      BNEQ    8$                                  ;
            03        00      08   A4  8F 00027          CASEB   8(R4), #0, #3                       ; 1513
   0026     0017    0008         0042     0002C 3$:      .WORD   7$-3$,-                             ;
                                                                 4$-3$,-                             ;
                                                                 5$-3$,-                             ;
                                                                 6$-3$                               ;
                      7E      0A   A2  3C 00034 4$:      MOVZWL  10(EVENT_DATA), -(SP)               ; 1517
                                       53  DD 00038      PUSHL   EVENT                               ;
                                       54  DD 0003A      PUSHL   R4                                  ;
                0000V  CF          03  FB 0003C          CALLS   #3, OUTPUT_CONSOLE                  ;
                                       3E  11 00041      BRB     8$                                  ; 1516
                      7E      0A   A2  3C 00043 5$:      MOVZWL  10(EVENT_DATA), -(SP)               ; 1519
                                       53  DD 00047      PUSHL   EVENT                               ;
                                       54  DD 00049      PUSHL   R4                                  ;
                0000V  CF          03  FB 0004B          CALLS   #3, OUTPUT_FILE                     ;
                                       2F  11 00050      BRB     8$                                  ; 1518
                      7E      0A   A2  3C 00052 6$:      MOVZWL  10(EVENT_DATA), -(SP)               ; 1523
                                       53  DD 00056      PUSHL   EVENT                               ;
                                       54  DD 00058      PUSHL   R4                                  ;
                0000V  CF          03  FB 0005A          CALLS   #3, OUTPUT_OPCOM                    ;
                      7E      0A   A2  3C 0005F          MOVZWL  10(EVENT_DATA), -(SP)               ; 1525
                                       53  DD 00063      PUSHL   EVENT                               ;
                                       54  DD 00065      PUSHL   R4                                  ;
                0000V  CF          03  FB 00067          CALLS   #3, OUTPUT_MONITOR                  ;
                                       13  11 0006C      BRB     8$                                  ; 1513
                      7E      08   A4  9A 0006E 7$:      MOVZBL  8(R4), -(SP)                        ; 1527
                                       01  DD 00072      PUSHL   #1                                  ;
                       00000000G   8F  DD 00074          PUSHL   #EVL$_BADTYPE                       ;
```

```
           00000000G  00            03  FB 0007A          CALLS   #3, LIB$SIGNAL              ;
                                    52  DD 00081  8$:     PUSHL   EVENT_DATA                  ; 1530
               0000G  CF            01  FB 00083          CALLS   #1, EVL$DEALLOCDBK          ;
                                    87  11 00088          BRB     2$                          ; 1505
                      50      04    AC  D0 0008A  9$:     MOVL    SINK, R0                    ; 1533
                      01      08    A0  91 0008E          CMPB    8(R0), #1                   ;
                                    0B  12 00092          BNEQ    10$                         ;
                              22    A0  D5 00094          TSTL    34(R0)                      ; 1534
                                    06  13 00097          BEQL    10$                         ;
                                    50  DD 00099          PUSHL   R0                          ; 1536
                 22   B0            01  FB 0009B          CALLS   #1, @34(R0)                 ;
                                        04 0009F  10$:    RET                                 ; 1539

; Routine Size:  160 bytes,    Routine Base:  $CODE$ + 0623
```

```
1 ROUTINE output_console (sink, event, event_length): NOVALUE =
1
1 !---
1 !
1 !       Output an event record to the console sink.
1 !
1 ! Inputs:
1 !
1 !       sink = Address of sink descriptor block
1 !       event = Address of event record
1 !       event_length = Length of event record
1 !
1 ! Outputs:
1 !
1 !       None
1 !---
1
2 BEGIN
2
2 MAP
2     sink:         REF BBLOCK,               ! Address of sink block
2     event:        REF BBLOCK;               ! Address of event record
2
2 LOCAL
2     status,
2     desc:         VECTOR [2];               ! Descriptor
2
2 IF .sink [sink$l_closertn] EQL 0            ! If sink not yet initialized,
2 THEN
3     BEGIN
3     IF .sink [sink$b_namelen] EQL 0         ! If no sink name specified,
3     THEN
4         BEGIN
4         sink [sink$b_namelen] = 6;          ! then default to _OPA0:
4         CH$MOVE(6, UPLIT BYTE('_OPA0:'), sink [sink$t_name]);
3         END;
3     open_file(.sink);                       ! then open the file/device
3     IF .sink [sink$l_closertn] EQL 0        ! If unsuccessful,
3     THEN
3         RETURN;                             ! then forget about the output
2     END;
2
2 format_event(.event, .event_length, write_line, .sink); ! Format the event
2
2 desc [0] = 0;
2 write_line(.sink, desc);                    ! write a null line afterwards
2
1 END;


                                               .PSECT  $PLIT$,NOWRT,NOEXE,2

                  3A  30  41  50  4F  5F  00084 P.AAQ:  .ASCII  \_OPA0:\                         ;
```

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

                                    007C 00000 OUTPUT_CONSOLE:
                                                        .WORD   Save R2,R3,R4,R5,R6                      ; 1540
                        5E          08  C2 00002        SUBL2   #8, SP                                   ;
                        56    04    AC  D0 00005        MOVL    SINK, R6                                 ; 1567
                              22    A6  D5 00009        TSTL    34(R6)                                   ;
                                    1C  12 0000C        BNEQ    2$                                       ;
                              2E    A6  95 0000E        TSTB    46(R6)                                   ; 1570
                                    0B  12 00011        BNEQ    1$                                       ;
                  2E    A6          06  90 00013        MOVB    #6, 46(R6)                               ; 1573
     2F    A6   0000'   CF          06  28 00017        MOVC3   #6, P.AAQ, 47(R6)                        ; 1574
                                    56  DD 0001E 1$:    PUSHL   R6                                       ; 1576
                0000V   CF          01  FB 00020        CALLS   #1, OPEN_FILE                            ;
                              22    A6  D5 00025        TSTL    34(R6)                                   ; 1577
                                    1A  13 00028        BEQL    3$                                       ;
                                    56  DD 0002A 2$:    PUSHL   R6                                       ; 1582
                            0000V   CF  9F 0002C        PUSHAB  WRITE_LINE                               ;
                        7E    08    AC  7D 00030        MOVQ    EVENT, -(SP)                             ;
                0000G   CF          04  FB 00034        CALLS   #4, FORMAT_EVENT                         ;
                                    6E  D4 00039        CLRL    DESC                                     ; 1584
                            4040    8F  BB 0003B        PUSHR   #^M<R6,SP>                               ; 1585
                0000V   CF          02  FB 0003F        CALLS   #2, WRITE_LINE                           ;
                                        04 00044 3$:    RET                                              ; 1587

; Routine Size:  69 bytes,    Routine Base:  $CODE$ + 06C3
```

```
1 ROUTINE write_line (sink, bufdesc): NOVALUE =
1
1 !---
1 !
1 !       This routine is called at the end of every line
1 !       in the formatted display.  The line is written
1 !       to the device or file specified by the sink.
1 !       If we are logging received events to the batch
1 !       job log, write the formatted text to the log as well.
1 !
1 ! Inputs:
1 !
1 !       sink = Address of sink control block
1 !       bufdesc = Address of descriptor of current output buffer
1 !
1 ! Outputs:
1 !
1 !       The line is written.
1 !---
1
2 BEGIN
2
2 MAP
2     sink:         REF BBLOCK;                 ! Address of sink control block
2
2 BIND
2     desc = .bufdesc:      VECTOR,             ! Address line descriptor
2     timeout = UPLIT(-15*10*1000*1000,-1); ! I/O timeout = 15 seconds
2
2 LOCAL
2     status,
2     fab:          REF BBLOCK,                 ! Address of FAB
2     rab:          REF BBLOCK;                 ! Address of RAB
2
2 IF .evl$gl_logmask [elg$v_rcvevt]            ! If logging received events,
2 THEN
2     LIB$PUT_OUTPUT(desc);                     ! then write line to batch job log
2
2 IF .sink [sink$l_closertn] EQL 0             ! If file no longer open,
2 THEN
2     RETURN;                                   ! then skip it
2
2 rab = .sink [sink$l_rab];                     ! Get address of RAB
2 fab = .rab [rab$l_fab];                       ! Get address of FAB
2
2 rab [rab$w_rsz] = .desc [0];                  ! Set up length/address of this line
2 rab [rab$l_rbf] = .desc [1];
2
2 status = $PUT(RAB = .rab);                    ! Output the record
2
2 IF NOT .status                               ! If error detected,
2 THEN
3     BEGIN
3     LOCAL name: VECTOR [2];
3     name [0] = .sink [sink$b_namelen];
3     name [1] = sink [sink$t_name];
3     SIGNAL(msg(writefile), T, name, .status, .rab [rab$l_stv]);
```

```
; 1077          1645  3      sink [sink$v_error] = true;               ! Suspend all operations on sink
; 1078          1646  3                                                ! until data base change
; 1079          1647  2      END;
; 1080          1648  2
; 1081          1649  1 END;


                                                         .PSECT  $PLIT$,NOWRT,NOEXE,2

                                             0008A       .BLKB   2
                     FFFFFFFF  F70F2E80      0008C P.AAR: .LONG  -150000000, -1                         ;

                                                 TIMEOUT=           P.AAR
                                                         .EXTRN  SYS$PUT, EVL$_WRITEFILE

                                                         .PSECT  $CODE$,NOWRT,2

                                       001C  00000 WRITE_LINE:
                                                         .WORD   Save R2,R3,R4                        ; 1588
                  5E          08  C2  00002              SUBL2   #8, SP
                  54     08   AC  D0  00005              MOVL    BUFDESC, R4                          ; 1614
   09      0000G  CF          06  E1  00009              BBC     #6, EVL$GL_LOGMASK, 1$               ; 1622
                              54  DD  0000F              PUSHL   R4                                   ; 1624
       00000000G  00          01  FB  00011              CALLS   #1, LIB$PUT_OUTPUT                   ;
                  53     04   AC  D0  00018 1$:          MOVL    SINK, R3                             ; 1626
                         22   A3  D5  0001C              TSTL    34(R3)                               ;
                              41  13  0001F              BEQL    2$                                   ;
                  52     1E   A3  D0  00021              MOVL    30(R3), RAB                          ; 1630
                  50     3C   A2  D0  00025              MOVL    60(RAB), FAB                         ; 1631
             22   A2          64  B0  00029              MOVW    (R4), 34(RAB)                        ; 1633
             28   A2     04   A4  D0  0002D              MOVL    4(R4), 40(RAB)                       ; 1634
                              52  DD  00032              PUSHL   RAB                                  ; 1636
       00000000G  00          01  FB  00034              CALLS   #1, SYS$PUT                          ;
                  24          50  E8  0003B              BLBS    STATUS, 2$                           ; 1638
                  6E     2E   A3  9A  0003E              MOVZBL  46(R3), NAME                         ; 1642
             04   AE     2F   A3  9E  00042              MOVAB   47(R3), NAME+4                       ; 1643
                         0C   A2  DD  00047              PUSHL   12(RAB)                              ; 1644
                              50  DD  0004A              PUSHL   STATUS                               ;
                         08   AE  9F  0004C              PUSHAB  NAME                                 ;
                              01  DD  0004F              PUSHL   #1                                   ;
                  00000000G   8F  DD  00051              PUSHL   #EVL$_WRITEFILE                      ;
       00000000G  00          05  FB  00057              CALLS   #5, LIB$SIGNAL                       ;
             14   A3          02  88  0005E              BISB2   #2, 20(R3)                           ; 1645
                              04  00062 2$:              RET                                          ; 1649

; Routine Size:  99 bytes,    Routine Base:  $CODE$ + 0708
```

```
1 ROUTINE output_file (sink, event, event_length): NOVALUE =
1
1 !---
1 !
1 !       Output an event record to a file.
1 !
1 ! Inputs:
1 !
1 !       sink = Address of sink descriptor block
1 !       event = Address of event record
1 !       event_length = Length of event record
1 !
1 ! Outputs:
1 !
1 !       None
1 !---
1
2 BEGIN
2
2 MAP
2     sink:          REF BBLOCK,               ! Address of sink block
2     event:         REF BBLOCK;               ! Address of event record
2
2 LOCAL
2     status,
2     name:          VECTOR [2],               ! Descriptor of file name
2     rab:           REF BBLOCK;               ! Address of RAB
2
2 IF .sink [sink$v_error]                      ! If in error state,
2 THEN
2     RETURN;                                  ! then don't do anything
2
2 name [0] = .sink [sink$b_namelen];           ! Setup descriptor of file name
2 name [1] = sink [sink$t_name];
2
2 IF .sink [sink$l_closertn] EQL 0             ! If sink is not yet initialized
2 THEN
2     open_file(.sink);                        ! then open the file
2
2 rab = .sink [sink$l_rab];                    ! Get address of RAB
2 rab [rab$w_rsz] = .event_length;             ! Set length of event record
2 rab [rab$l_rbf] = .event;                    ! and address
2
2 status = $PUT(RAB = .rab);                   ! Write event record to file
2
2 IF NOT .status                               ! If error detected,
2 THEN
3     BEGIN
3     SIGNAL(msg(writefile), 1, name, .status, .rab [rab$l_stv]);
3     close_file(.sink);                       ! Close sink file
3     sink [sink$v_error] = true;              ! Suspend all operations on sink
3                                              ! until data base change
2     END;
2
1 END;
```

```
                                        000C 00000 OUTPUT_FILE:
                                                           .WORD    Save R2,R3                           ; 1650
                    5E             08  C2 00002            SUBL2    #8, SP
                    53       04    AC  D0 00005            MOVL     SINK, R3                             ; 1678
     51      14     A3             01  E0 00009            BBS      #1, 20(R3), 2$
                    6E       2E    A3  9A 0000E            MOVZBL   46(R3), NAME                         ; 1682
             04     AE       2F    A3  9E 00012            MOVAB    47(R3), NAME+4                       ; 1683
                             22    A3  D5 00017            TSTL     34(R3)                               ; 1685
                                   07  12 0001A            BNEQ     1$
                                   53  DD 0001C            PUSHL    R3                                   ; 1687
          0000V     CF             01  FB 0001E            CALLS    #1, OPEN_FILE
                    52       1E    A3  D0 00023 1$:        MOVL     30(R3), RAB                          ; 1689
             22     A2       0C    AC  B0 00027            MOVW     EVENT_LENGTH, 34(RAB)                ; 1690
             28     A2       08    AC  D0 0002C            MOVL     EVENT, 40(RAB)                       ; 1691
                                   52  DD 00031            PUSHL    RAB                                  ; 1693
      00000000G     00             01  FB 00033            CALLS    #1, SYS$PUT
                    22             50  E8 0003A            BLBS     STATUS, 2$                           ; 1695
                             0C    A2  DD 0003D            PUSHL    12(RAB)                              ; 1698
                                   50  DD 00040            PUSHL    STATUS
                             08    AE  9F 00042            PUSHAB   NAME
                                   01  DD 00045            PUSHL    #1
                     00000000G     8F  DD 00047            PUSHL    #EVL$_WRITEFILE
      00000000G     00             05  FB 0004D            CALLS    #5, LIB$SIGNAL
                                   53  DD 00054            PUSHL    R3                                   ; 1699
          0000V     CF             01  FB 00056            CALLS    #1, CLOSE_FILE
             14     A3             02  88 0005B            BISB2    #2, 20(R3)                           ; 1700
                                       04 0005F 2$:        RET                                           ; 1704

; Routine Size:  96 bytes,    Routine Base:  $CODE$ + 076B
```

```
GLOBAL ROUTINE open_file (sink): NOVALUE =

!---
!
!       Open a file sink.
!
! Inputs:
!
!       sink = Address of sink control block
!
! Outputs:
!
!       None
!---

BEGIN

MAP
    sink:       REF BBLOCK;             ! Address of sink block

LOCAL
    status,
    length,                             ! Length of FAB and RAB storage
    name:       VECTOR [2],             ! Descriptor of sink name
    rab:        REF BBLOCK,             ! Address of RAB
    fab:        REF BBLOCK;             ! Address of FAB

name [0] = .sink [sink$b_namelen];      ! Setup descriptor of file name
name [1] = sink [sink$t_name];

length = rab$c_bln + fab$c_bln;
signal_if_error(LIB$GET_VM(length,rab));        ! Allocate RAB/FAB

fab = .rab + rab$c_bln;                 ! Get address of FAB

$FAB_INIT(FAB = .fab,                   ! Initialize FAB
        FNS = .name [0],
        FNA = .name [1],
        DNM = '.LOG',
        FAC = PUT,
        RAT = CR,
        FOP = (CIF,SQO),
        SHR = (GET,UPI));
$RAB_INIT(RAB = .rab,                   ! Initialize RAB
        FAB = .fab,
        ROP = (EOF,WBH));

status = $CREATE(FAB = .fab);           ! Create the event file

IF NOT .status                          ! If error detected,
THEN
    BEGIN
    IF .fab [fab$l_stv] NEQ ss$_devalloc ! If not "device allocated" error,
    THEN                                ! (which is ok, since user might want
                                        ! to use the output terminal)
        SIGNAL(msg(openfile), 1, name, .status, .fab [fab$l_stv]);
    LIB$FREE_VM(length,rab);            ! Deallocate storage
```

```
; 1196       1762 3      RETURN;
; 1197       1763 2      END;
; 1198       1764 2
; 1199       1765 2 status = $CONNECT(RAB = .rab);            ! Connect to stream
; 1200       1766 2 IF NOT .status                            ! If error detected,
; 1201       1767 2 THEN
; 1202       1768 3      BEGIN
; 1203       1769 3      SIGNAL(msg(openfile), 1, name, .status, .rab [rab$l_stv]);
; 1204       1770 3      LIB$FREE_VM(length,rab);             ! Deallocate storage
; 1205       1771 3      RETURN;
; 1206       1772 2      END;
; 1207       1773 2
; 1208       1774 2 sink [sink$l_rab] = .rab;                 ! Save address of RAB
; 1209       1775 2 sink [sink$l_closertn] = close_file;     ! Set file opened
; 1210       1776 2
; 1211       1777 1 END;


                                                      .PSECT  $PLIT$,NOWRT,NOEXE,2

                     47  4F  4C  2E  00094 P.AAS:     .ASCII  \.LOG\                                          ;

                                                      .EXTRN  SYS$CREATE, EVL$_OPENFILE
                                                      .EXTRN  SYS$CONNECT

                                                      .PSECT  $CODE$,NOWRT,2

                                   07FC 00000         .ENTRY  OPEN_FILE, Save R2,R3,R4,R5,R6,R7,R8,R9,R10 ; 1705
                   5A 00000000G  8F  D0 00002         MOVL    #EVL$_OPENFILE, R10                         ;
                   59 00000000G  00  9E 00009         MOVAB   LIB$SIGNAL, R9                              ;
                   5E            10  C2 00010         SUBL2   #16, SP                                     ;
                   58        04  AC  D0 00013         MOVL    SINK, R8                                    ; 1732
              08   AE        2E  A8  9A 00017         MOVZBL  46(R8), NAME                                ;
              0C   AE        2F  A8  9E 0001C         MOVAB   47(R8), NAME+4                              ; 1733
              04   AE        94  8F  9A 00021         MOVZBL  #148, LENGTH                                ; 1735
                             5E  DD 00026             PUSHL   SP                                          ; 1736
                         08  AE  9F 00028             PUSHAB  LENGTH                                      ;
           00000000G  00     02  FB 0002B             CALLS   #2, LIB$GET_VM                              ;
                      52     50  D0 00032             MOVL    R0, STATUS                                  ;
                      06     52  E8 00035             BLBS    STATUS, 1$                                  ;
                             52  DD 00038             PUSHL   STATUS                                      ;
                      69     01  FB 0003A             CALLS   #1, LIB$SIGNAL                              ;
                                 04 0003D             RET                                                 ;
                      57     6E  D0 0003E 1$:         MOVL    RAB, R7                                     ; 1738
                      56  44 A7  9E 00041             MOVAB   68(R7), FAB                                 ;
  0050   8F        00  6E    00  2C 00045             MOVC5   #0, (SP), #0, #80, (FAB)                    ; 1747
                             66     0004C                                                                 ;
                  66  5003   8F  B0 0004D             MOVW    #20483, (FAB)                               ;
              04  A6 02000040 8F D0 00052             MOVL    #33554496, 4(FAB)                           ;
              16  A6  4201   8F  B0 0005A             MOVW    #16897, 22(FAB)                             ;
              1E  A6  0202   8F  B0 00060             MOVW    #514, 30(FAB)                               ;
              2C  A6    0C   AE  D0 00066             MOVL    NAME+4, 44(FAB)                             ;
              30  A6  0000'  CF  9E 0006B             MOVAB   P.AAS, 48(FAB)                              ;
              34  A6    08   AE  90 00071             MOVB    NAME, 52(FAB)                               ;
              35  A6         04  90 00076             MOVB    #4, 53(FAB)                                 ;
  0044   8F        00  6E    00  2C 0007A             MOVC5   #0, (SP), #0, #68, (R7)                     ; 1750
```

```
                                 67          00081                                             ;
                  67     4401    8F       B0 00082          MOVW     #17409, (R7)              ;
            04    A7     0500    8F       3C 00087          MOVZWL   #1280, 4(R7)              ;
            3C    A7             56       D0 0008D          MOVL     FAB, 60(R7)               ;
                                 56       DD 00091          PUSHL    FAB                       ; 1752
     00000000G    00             01       FB 00093          CALLS    #1, SYS$CREATE            ;
                  53             50       D0 0009A          MOVL     R0, STATUS                ;
                  0F             53       E8 0009D          BLBS     STATUS, 2$                ; 1754
     00000840     8F       0C    A6       D1 000A0          CMPL     12(FAB), #2112            ; 1757
                                 26       13 000A8          BEQL     4$                        ;
                           0C    A6       DD 000AA          PUSHL    12(FAB)                   ; 1760
                                 15       11 000AD          BRB      3$                        ;
                  52             6E       D0 000AF  2$:     MOVL     RAB, R2                   ; 1765
                                 52       DD 000B2          PUSHL    R2                        ;
     00000000G    00             01       FB 000B4          CALLS    #1, SYS$CONNECT           ;
                  53             50       D0 000BB          MOVL     R0, STATUS                ;
                  1C             53       E8 000BE          BLBS     STATUS, 5$                ; 1766
                           0C    A2       DD 000C1          PUSHL    12(R2)                    ; 1769
                                 53       DD 000C4  3$:     PUSHL    STATUS                    ;
                           10    AE       9F 000C6          PUSHAB   NAME                      ;
                                 01       DD 000C9          PUSHL    #1                        ;
                                 5A       DD 000CB          PUSHL    R10                       ;
                  69             05       FB 000CD          CALLS    #5, LIB$SIGNAL            ;
                                 5E       DD 000D0  4$:     PUSHL    SP                        ; 1770
                           08    AE       9F 000D2          PUSHAB   LENGTH                    ;
     00000000G    00             02       FB 000D5          CALLS    #2, LIB$FREE_VM           ;
                                          04 000DC          RET                                ; 1768
            1E    A8             6E       D0 000DD  5$:     MOVL     RAB, 30(R8)               ; 1774
            22    A8    0000V    CF       9E 000E1          MOVAB    CLOSE_FILE, 34(R8)        ; 1775
                                          04 000E7          RET                                ; 1777

; Routine Size:  232 bytes,    Routine Base:  $CODE$ + 07CB
```

```
1 ROUTINE close_file (sink): NOVALUE =
1
1 !---
1 !
1 !       Close a file sink.
1 !
1 ! Inputs:
1 !
1 !       sink = Address of sink block
1 !
1 ! Outputs:
1 !
1 !       None
1 !---
1
2 BEGIN
2
2 MAP
2     sink:       REF BBLOCK;             ! Address of sink block
2
2 LOCAL
2     status,
2     rab:        REF BBLOCK;             ! Address of RAB
2
2 rab = .sink [sink$l_rab];               ! Get address of RAB
2
2 IF .rab NEQ 0                           ! If RAB and FAB were allocated
2 THEN
3     BEGIN
3     LOCAL
3         length,
3         name:   VECTOR [2],             ! Descriptor of file name
3         fab:    REF BBLOCK;             ! Address of FAB
3
3     fab = .rab [rab$l_fab];             ! Get address of FAB
3
3     name [0] = .sink [sink$b_namelen];  ! Setup descriptor of file name
3     name [1] = sink [sink$t_name];
3
3     $DISCONNECT(RAB = .rab);            ! Disconnect to cancel any I/O
3
3     status = $CLOSE(FAB = .fab);        ! Close the file
3
3     IF NOT .status                      ! If error detected,
3     THEN
3         SIGNAL(msg(closefile), 1, name, .status, .fab [fab$l_stv]);
3
3     length = rab$c_bln + fab$c_bln;     ! Length of RAB/FAB storage
3     LIB$FREE_VM(length,rab);            ! Deallocate RAB and FAB
2     END;
2
2 sink [sink$l_closertn] = 0;             ! Set sink no longer open
2
1 END;
```

```
                                                            .EXTRN   SYS$DISCONNECT, SYS$CLOSE
                                                            .EXTRN   EVL$_CLOSEFILE

                                   000C 00000 CLOSE_FILE:
                                                            .WORD    Save R2,R3                    ; 1778
              5E               0C   C2 00002                SUBL2    #12, SP                       ;
              52        04     AC   D0 00005                MOVL     SINK, R2                      ; 1802
                        1E     A2   DD 00009                PUSHL    30(R2)                        ;
              50               6E   D0 0000C                MOVL     RAB, R0                       ; 1804
                               4B   13 0000F                BEQL     2$                            ;
              53        3C     A0   D0 00011                MOVL     60(R0), FAB                   ; 1812
         08   AE        2E     A2   9A 00015                MOVZBL   46(R2), NAME                  ; 1814
         0C   AE        2F     A2   9E 0001A                MOVAB    47(R2), NAME+4                ; 1815
                               50   DD 0001F                PUSHL    R0                            ; 1817
    00000000G  00              01   FB 00021                CALLS    #1, SYS$DISCONNECT            ;
                               53   DD 00028                PUSHL    FAB                           ; 1819
    00000000G  00              01   FB 0002A                CALLS    #1, SYS$CLOSE                 ;
              17               50   E8 00031                BLBS     STATUS, 1$                    ; 1821
                        0C     A3   DD 00034                PUSHL    12(FAB)                       ; 1823
                               50   DD 00037                PUSHL    STATUS                        ;
                        10     AE   9F 00039                PUSHAB   NAME                          ;
                               01   DD 0003C                PUSHL    #1                            ;
                   00000000G   8F   DD 0003E                PUSHL    #EVL$_CLOSEFILE               ;
    00000000G  00              05   FB 00044                CALLS    #5, LIB$SIGNAL                ;
         04   AE        94     8F   9A 0004B 1$:            MOVZBL   #148, LENGTH                  ; 1825
                               5E   DD 00050                PUSHL    SP                            ; 1826
                        08     AE   9F 00052                PUSHAB   LENGTH                        ;
    00000000G  00              02   FB 00055                CALLS    #2, LIB$FREE_VM               ;
                        22     A2   D4 0005C 2$:            CLRL     34(R2)                        ; 1829
                                    04 0005F                RET                                    ; 1831

; Routine Size:  96 bytes,    Routine Base:  $CODE$ + 08B3
```

```
1 ROUTINE output_opcom (sink, event, event_length): NOVALUE =
1
1 !---
1 !
1 !       Output a formatted event to all OPCOM network operators
1 !
1 ! Inputs:
1 !
1 !       sink = Address of sink descriptor block
1 !       event = Address of event record
1 !       event_length = Length of event record
1 !
1 ! Outputs:
1 !
1 !       None
1 !---
1
2 BEGIN
2
2 MAP
2     sink:       REF BBLOCK,               ! Address of sink block
2     event:      REF BBLOCK;               ! Address of event record
2
2 LOCAL
2     status,
2     opcom_desc: VECTOR [2],               ! Descriptor of OPCOM buffer
2     opcom_buf:  BBLOCK [$BYTEOFFSET(opc$l_ms_text)+2048];
2
2 !
2 ! Always write the formatted event to all OPCOM network operators,
2 ! whether a monitor task is specified or not.  This allows monitor
2 ! tasks to receive and process events, and we still get the OPCOM feature.
2 !
2
2 sink [sink$w_maxbufsiz] = 2048;           ! Set maximum limit for output
2 sink [sink$w_buflen] = 0;                 ! Initialize buffer descriptor
2 sink [sink$l_buffer] = opcom_buf + $BYTEOFFSET(opc$l_ms_text);
2
2 format_event(.event, .event_length,       ! Call formatting routines
2             save_lines, .sink);           ! but append output lines together
2
2 sink [sink$w_buflen] = .sink [sink$w_buflen] - 2;        ! Remove trailing CR/LF
2
2 opcom_buf [opc$b_ms_type] = opc$_rq_rqst;         ! REQUEST to be sent to operator
2 opcom_buf [opc$b_ms_target] = opc$m_nm_ntwork;    ! Sent to network operators
2 opcom_buf [opc$w_ms_status] = 0;                  ! No status
2 opcom_buf [opc$l_ms_rqstid] = 0;                  ! No request ID
2
2 opcom_desc [0] = .sink [sink$w_buflen] + $BYTEOFFSET(opc$l_ms_text);
2 opcom_desc [1] = opcom_buf;
2
2 status = $SNDOPR(MSGBUF = opcom_desc);   ! Send message to OPCOM
2
2 IF NOT .status                            ! If error detected,
2 THEN
2     SIGNAL(msg(brdcst), 1, %ASCID 'OPCOM', .status); ! then signal the error
2
```

```
; 1325          1889  1 END;


                                                                   .PSECT   $PLIT$,NOWRT,NOEXE,2

                00  00  00  4D  4F  43  50  4F   00098 P.AAU:      .ASCII   \OPCOM\<0><0><0>                              ;
                                      010E0005   000A0 P.AAT:      .LONG    17694725                                      ;
                                      00000000'  000A4             .ADDRESS P.AAU                                         ;

                                                                   .EXTRN   SYS$SNDOPR, EVL$_BRDCST

                                                                   .PSECT   $CODE$,NOWRT,2

                                          0004 00000 OUTPUT_OPCOM:
                                                                   .WORD    Save R2                                       ; 1832
                    5E        F7F0   CE   9E 00002                 MOVAB    -2064(SP), SP                                 ;
                    52          04   AC   D0 00007                 MOVL     SINK, R2                                      ; 1866
              16    A2        0800   8F   3C 0000B                 MOVZWL   #2048, 22(R2)                                 ;
              1A    A2          08   AE   9E 00011                 MOVAB    OPCOM_BUF+8, 26(R2)                           ; 1868
                                     52   DD 00016                 PUSHL    R2                                            ; 1871
                             0000V   CF   9F 00018                 PUSHAB   SAVE_LINES                                    ; 1870
                    7E          08   AC   7D 0001C                 MOVQ     EVENT, -(SP)                                  ;
            0000G   CF               04   FB 00020                 CALLS    #4, FORMAT_EVENT                              ;
              18    A2               02   A2 00025                 SUBW2    #2, 24(R2)                                    ; 1873
                    6E        4003   8F   3C 00029                 MOVZWL   #16387, OPCOM_BUF                             ; 1875
                                04   AE   D4 0002E                 CLRL     OPCOM_BUF+4                                   ; 1878
              F8    AD          18   A2   3C 00031                 MOVZWL   24(R2), OPCOM_DESC                            ; 1880
              F8    AD               08   C0 00036                 ADDL2    #8, OPCOM_DESC                                ;
              FC    AD               6E   9E 0003A                 MOVAB    OPCOM_BUF, OPCOM_DESC+4                       ; 1881
                                     7E   D4 0003E                 CLRL     -(SP)                                         ; 1883
                                F8   AD   9F 00040                 PUSHAB   OPCOM_DESC                                    ;
        00000000G   00               02   FB 00043                 CALLS    #2, SYS$SNDOPR                                ;
                    15               50   E8 0004A                 BLBS     STATUS, 1$                                    ; 1885
                                     50   DD 0004D                 PUSHL    STATUS                                        ; 1887
                             0000'   CF   9F 0004F                 PUSHAB   P.AAT                                         ;
                                     01   DD 00053                 PUSHL    #1                                            ;
                         00000000G   8F   DD 00055                 PUSHL    #EVL$_BRDCST                                  ;
        00000000G   00               04   FB 0005B                 CALLS    #4, LIB$SIGNAL                                ;
                                          04 00062 1$:             RET                                                    ; 1889

; Routine Size:  99 bytes,    Routine Base:  $CODE$ + 0913
```

```
ROUTINE output_monitor (sink, event, event_length): NOVALUE =

!---
!
!       Output an event record to a monitor process.
!
! Inputs:
!
!       sink = Address of sink descriptor block
!       event = Address of event record
!       event_length = Length of event record
!
! Outputs:
!
!       None
!---

BEGIN

MAP
    sink:       REF BBLOCK,                 ! Address of sink block
    event:      REF BBLOCK;                 ! Address of event record

LOCAL
    status;

!
! If a monitor name is specified, then send the event to the remote task.
! We must check for non-null monitor name, since it is valid to have the
! monitor sink turned on without any name, meaning only OPCOM events.
!

IF .sink [sink$v_error]                     ! If in error state,
THEN
    RETURN;                                 ! then don't do anything

IF .sink [sink$b_namelen] NEQ 0             ! If monitor name specified,
THEN
    BEGIN
    IF .sink [sink$l_closertn] EQL 0        ! If sink is not yet initialized
    THEN
        open_monitor(.sink);                ! then connect to process

    status = $QIOW(FUNC = IO$_WRITEVBLK,   ! Send event to monitor process
                CHAN = .sink [sink$w_channel],
                EFN = evl$c_synch_efn,
                IOSB = sink [sink$w_iosb],
                P1 = .event,                ! Address of event record
                P2 = .event_length);        ! Length of event record

    IF .status                              ! If successfully submitted,
    THEN
        status = .sink [sink$w_iosb];       ! then get final I/O status

    IF NOT .status                          ! If error detected,
    THEN
        BEGIN
```

```
; 1384        1947  4          IF .status NEQ ss$_abort          ! If something other than link abort,
; 1385        1948  4          THEN
; 1386        1949  5              BEGIN                         ! then report error
; 1387        1950  5              LOCAL
; 1388        1951  5                  sink_name: VECTOR [2];
; 1389        1952  5              sink_name [0] = .sink [sink$b_namelen];
; 1390        1953  5              sink_name [1] = sink [sink$t_name];
; 1391        1954  5              SIGNAL(msg(writemon), 1, sink_name, .status);
; 1392        1955  4              END;
; 1393        1956  4          close_monitor(.sink);             ! Close the sink
; 1394        1957  4          sink [sink$v_error] = true;       ! Suspend all operations on sink
; 1395        1958  4                                            ! until data base change
; 1396        1959  3          END;
; 1397        1960  2      END;
; 1398        1961  2
; 1399        1962  1 END;


                                                                   .EXTRN  EVL$_WRITEMON

                                                  0004 00000 OUTPUT_MONITOR:
                                                                   .WORD   Save R2                          ; 1890
                       5E                08  C2 00002              SUBL2   #8, SP
                       52           04   AC  D0 00005              MOVL    SINK, R2                         ; 1922
        64        14   A2                01  E0 00009              BBS     #1, 20(R2), 4$
                                    2E   A2  95 0000E              TSTB    46(R2)                           ; 1926
                                         5F  13 00011              BEQL    4$
                                    22   A2  D5 00013              TSTL    34(R2)                           ; 1929
                                         07  12 00016              BNEQ    1$
                                         52  DD 00018              PUSHL   R2                               ; 1931
              0000V   CF                 01  FB 0001A              CALLS   #1, OPEN_MONITOR
                                         7E  7C 0001F 1$:          CLRQ    -(SP)                            ; 1938
                                         7E  7C 00021              CLRQ    -(SP)
                       7E           08   AC  7D 00023              MOVQ    EVENT, -(SP)
                                         7E  7C 00027              CLRQ    -(SP)
                                    26   A2  9F 00029              PUSHAB  38(R2)
                                         30  DD 0002C              PUSHL   #48
                       7E           1E   A2  3C 0002E              MOVZWL  30(R2), -(SP)
                                         01  DD 00032              PUSHL   #1
          00000000G   00                 0C  FB 00034              CALLS   #12, SYS$QIOW
                       07                50  E9 0003B              BLBC    STATUS, 2$                       ; 1940
                       50           26   A2  3C 0003E              MOVZWL  38(R2), STATUS                   ; 1942
                       2D                50  E8 00042              BLBS    STATUS, 4$                       ; 1944
                       2C                50  D1 00045 2$:          CMPL    STATUS, #44                      ; 1947
                                         1D  13 00048              BEQL    3$
                       6E           2E   A2  9A 0004A              MOVZBL  46(R2), SINK_NAME                ; 1952
                  04   AE           2F   A2  9E 0004E              MOVAB   47(R2), SINK_NAME+4              ; 1953
                                         50  DD 00053              PUSHL   STATUS                           ; 1954
                                    04   AE  9F 00055              PUSHAB  SINK_NAME
                                         01  DD 00058              PUSHL   #1
                           00000000G     8F  DD 0005A              PUSHL   #EVL$_WRITEMON
          00000000G   00                 04  FB 00060              CALLS   #4, LIB$SIGNAL
                                         52  DD 00067 3$:          PUSHL   R2                               ; 1956
              0000V   CF                 01  FB 00069              CALLS   #1, CLOSE_MONITOR
                  14   A2                02  88 0006E              BISB2   #2, 20(R2)                       ; 1957
                                             04 00072 4$:          RET                                      ; 1962
```

```
; Routine Size:  115 bytes,    Routine Base:  $CODE$ + 0976
```

```
  1 ROUTINE save_lines (sink, desc, short_form): NOVALUE =
  1
  1 !---
  1 !
  1 !       This is an action routine called by the event formatting
  1 !       routines for each line of output.  It appends each line
  1 !       into an buffer associated with the sink followed by a CR/LF.
  1 !
  1 ! Inputs:
  1 !
  1 !       sink = Address of sink block
  1 !       desc = Address of descriptor of formatted line
  1 !       short_form = Length of line if short form is desired
  1 !                     (-1 if no short form possible)
  1 !
  1 ! Outputs:
  1 !
  1 !       The line is appended to the buffer
  1 !---
  1
  2 BEGIN
  2
  2 MAP
  2     sink:       REF BBLOCK,                 ! Address of sink control block
  2     desc:       REF VECTOR;                 ! Descriptor of output line
  2
  2 LOCAL
  2     linedesc:   VECTOR [2],                 ! Descriptor of the line
  2     outdesc:    VECTOR [2];                 ! Output buffer descriptor
  2
  2 outdesc [0] = .sink [sink$w_maxbufsiz] - .sink [sink$w_buflen];
  2 outdesc [1] = .sink [sink$l_buffer] + .sink [sink$w_buflen];
  2
  2 linedesc [0] = .desc [0];                   ! Copy descriptor locally
  2 linedesc [1] = .desc [1];
  2
  2 IF .short_form GTR 0                        ! If there is a short form,
  2 THEN
  2     linedesc [0] = .short_form              ! then use that because OPCOM
  2 ELSE                                        ! is limited in output size
  2 IF .short_form EQL 0                        ! If short form means no output,
  2 THEN
  2     RETURN;                                 ! then don't append anything at all
  2
P 2 $FAO(%ASCID '!AS!/',                        ! Append the string to the buffer
P 2         outdesc,outdesc,                    ! Result buffer/length
  2         linedesc);                          ! Parameter is descriptor of line
  2
  2 sink [sink$w_buflen] = .sink [sink$w_buflen] + .outdesc [0]; ! Update length
  2
  1 END;


                                                      .PSECT  $PLIT$,NOWRT,NOEXE,2

        00  00  00  2F  21  53  41  21  000A8 P.AAW:  .ASCII  \!AS!/\<0><0><0>                    ;
```

```
                              010E0005  000B0 P.AAV:   .LONG    17694725                          ;
                              00000000' 000B4          .ADDRESS P.AAW                             ;

                                                       .EXTRN   SYS$FAO

                                                       .PSECT   $CODE$,NOWRT,2

                                   0004 00000 SAVE_LINES:
                                                       .WORD    Save R2                           ; 1963
                    5E        0C    C2 00002           SUBL2    #12, SP                           ;
                    52    04  AC    D0 00005           MOVL     SINK, R2                          ; 1993
                    50    16  A2    3C 00009           MOVZWL   22(R2), R0                        ;
                    51    18  A2    3C 0000D           MOVZWL   24(R2), R1                        ;
          7E        50        51    C3 00011           SUBL3    R1, R0, OUTDESC                   ;
                    50    18  A2    3C 00015           MOVZWL   24(R2), R0                        ; 1994
               04   AE  1A B240     9E 00019           MOVAB    @26(R2)[R0], OUTDESC+4            ;
                    50    08  AC    D0 0001F           MOVL     DESC, R0                          ; 1996
               08   AE        60    7D 00023           MOVQ     (R0), LINEDESC                    ;
                    50    0C  AC    D0 00027           MOVL     SHORT_FORM, R0                    ; 1999
                              06    15 0002B           BLEQ     1$                                ;
               08   AE        50    D0 0002D           MOVL     R0, LINEDESC                      ; 2001
                              02    11 00031           BRB      2$                                ;
                              18    13 00033 1$:       BEQL     3$                                ; 2003
                          08  AE    9F 00035 2$:       PUSHAB   LINEDESC                          ; 2009
                          04  AE    9F 00038           PUSHAB   OUTDESC                           ;
                          08  AE    9F 0003B           PUSHAB   OUTDESC                           ;
                       0000'  CF    9F 0003E           PUSHAB   P.AAV                             ;
          00000000G 00        04    FB 00042           CALLS    #4, SYS$FAO                       ;
               18   A2        6E    A0 00049           ADDW2    OUTDESC, 24(R2)                   ; 2011
                              04  0004D 3$:            RET                                        ; 2013

; Routine Size:  78 bytes,    Routine Base:  $CODE$ + 09E9
```

```
ROUTINE open_monitor (sink): NOVALUE =

!---
!
!       Open the monitor process sink.
!
! Inputs:
!
!       sink = Address of sink control block
!
! Outputs:
!
!       None
!---

BEGIN

MAP
    sink:       REF BBLOCK;             ! Address of sink block

LOCAL
    status,
    ncb_desc: VECTOR [2],               ! Descriptor of NCB for process
    ncb:        VECTOR [128,BYTE];      ! NCB

status = $ASSIGN(DEVNAM = %ASCID '_NET:',
                CHAN = sink [sink$w_channel]);

IF NOT .status                  ! If error detected,
THEN
    (SIGNAL(msg(netasn), 0, .status); RETURN);  ! then report error

CH$COPY(CH$RCHAR(evl$gt_localnode+2), evl$gt_localnode+3,
        8, UPLIT('::"TASK='),
        .sink [sink$b_namelen], sink [sink$t_name],
        1, UPLIT('"'),
        0, 128, ncb);

ncb_desc [0] = CH$RCHAR(evl$gt_localnode+2)+8+.sink [sink$b_namelen]+1;
ncb_desc [1] = ncb;

status = $QIOW(FUNC = IO$_ACCESS,       ! Connect to monitor process
                CHAN = .sink [sink$w_channel],
                EFN = evl$c_synch_efn,
                IOSB = sink [sink$w_iosb],
                P2 = ncb_desc);

IF .status                              ! If successfully submitted,
THEN
    status = .sink [sink$w_iosb];       ! then get final I/O status

IF NOT .status                          ! If error detected,
THEN
    (SIGNAL(msg(openmon), 1, ncb_desc, .status); RETURN);

sink [sink$l_closertn] = close_monitor; ! Set connection established
```

```
2 IF .evl$gl_logmask [elg$v_monopn]          ! If logging monitor connections,
2 THEN
2     SIGNAL(msg(logopnm), 3, .ncb_desc [0], .ncb_desc [1], 0);
2
1 END;


                                                         .PSECT  $PLIT$,NOWRT,NOEXE,2

  00 00 00 3A 54 45 4E 5F  000B8 P.AAY:  .ASCII  \_NET:\<0><0><0>                             ;
                  010E0005  000C0 P.AAX:  .LONG   17694725                                     ;
                 00000000'  000C4         .ADDRESS P.AAY                                       ;
  3D 4B 53 41 54 22 3A 3A  000C8 P.AAZ:  .ASCII  \::"TASK=\                                   ;
              00 00 00 22  000D0 P.ABA:  .ASCII  \"\<0><0><0>                                 ;

                                                         .EXTRN  EVL$_OPENMON, EVL$_LOGOPNM

                                                         .PSECT  $CODE$,NOWRT,2

                              OFFC 00000 OPEN_MONITOR:
                                                         .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 2014
                 5E     FF78    CE 9E 00002              MOVAB   -136(SP), SP                                ;
                                7E 7C 00007              CLRQ    -(SP)                                       ; 2040
                 56       04    AC D0 00009              MOVL    SINK, R6                                    ;
                          1E    A6 9F 0000D              PUSHAB  30(R6)                                      ;
                        0000'   CF 9F 00010              PUSHAB  P.AAX                                       ;
            00000000G   00      04 FB 00014              CALLS   #4, SYS$ASSIGN                              ;
                        5B      50 D0 0001B              MOVL    R0, STATUS                                  ;
                        12      5B E8 0001E              BLBS    STATUS, 1$                                  ; 2042
                                5B DD 00021              PUSHL   STATUS                                      ; 2044
                                7E D4 00023              CLRL    -(SP)                                       ;
                     00000000G  8F DD 00025              PUSHL   #EVL$_NETASN                                ;
            00000000G   00      03 FB 0002B              CALLS   #3, LIB$SIGNAL                              ;
                                   04 00032              RET                                                 ;
                 5A     0000G   CF 9A 00033 1$:          MOVZBL  EVL$GT_LOCALNODE+2, R10                     ; 2046
                 59       2E    A6 9A 00038              MOVZBL  46(R6), R9                                  ; 2048
                 58       80    8F 9A 0003C              MOVZBL  #128, R8                                    ;
                 57             6E 9E 00040              MOVAB   NCB, R7                                     ;
  58         00   0000G   CF    5A 2C 00043              MOVC5   R10, EVL$GT_LOCALNODE+3, #0, R8, (R7)       ;
                                67    0004A                                                                  ;
                                2D 18 0004B              BGEQ    2$                                          ;
                 57             5A C0 0004D              ADDL2   R10, R7                                     ;
                 58             5A C2 00050              SUBL2   R10, R8                                     ;
  58         00   0000'   CF    08 2C 00053              MOVC5   #8, P.AAZ, #0, R8, (R7)                     ;
                                67    0005A                                                                  ;
                                1D 18 0005B              BGEQ    2$                                          ;
                 57             08 C0 0005D              ADDL2   #8, R7                                      ;
                 58             08 C2 00060              SUBL2   #8, R8                                      ;
  58         00     2F    A6    59 2C 00063              MOVC5   R9, 47(R6), #0, R8, (R7)                    ;
                                67    00069                                                                  ;
                                0E 18 0006A              BGEQ    2$                                          ;
                 57             59 C0 0006C              ADDL2   R9, R7                                      ;
                 58             59 C2 0006F              SUBL2   R9, R8                                      ;
  58         00   0000'   CF    01 2C 00072              MOVC5   #1, P.ABA, #0, R8, (R7)                     ;
                                67    00079                                                                  ;
                 50     0000G   CF 9A 0007A 2$:          MOVZBL  EVL$GT_LOCALNODE+2, R0                      ; 2052
```

```
                  51        2E   A6  9A 0007F         MOVZBL  46(R6), R1                                   ;
                  50             51  C0 00083         ADDL2   R1, R0                                       ;
          F8      AD        09   A0  9E 00086         MOVAB   9(R0), NCB_DESC                              ;
          FC      AD             6E  9E 0008B         MOVAB   NCB, NCB_DESC+4                              ; 2053
                                 7E  7C 0008F         CLRQ    -(SP)                                        ; 2059
                                 7E  7C 00091         CLRQ    -(SP)                                        ;
                            F8   AD  9F 00093         PUSHAB  NCB_DESC                                     ;
                                 7E  7C 00096         CLRQ    -(SP)                                        ;
                                 7E  D4 00098         CLRL    -(SP)                                        ;
                            26   A6  9F 0009A         PUSHAB  38(R6)                                       ;
                                 32  DD 0009D         PUSHL   #50                                          ;
                  7E        1E   A6  3C 0009F         MOVZWL  30(R6), -(SP)                                ;
                                 01  DD 000A3         PUSHL   #1                                           ;
       00000000G  00             0C  FB 000A5         CALLS   #12, SYS$QIOW                                ;
                  5B             50  D0 000AC         MOVL    R0, STATUS                                   ;
                  07             5B  E9 000AF         BLBC    STATUS, 3$                                   ; 2061
                  5B        26   A6  3C 000B2         MOVZWL  38(R6), STATUS                               ; 2063
                  15             5B  E8 000B6         BLBS    STATUS, 4$                                   ; 2065
                                 5B  DD 000B9 3$:     PUSHL   STATUS                                       ; 2067
                            F8   AD  9F 000BB         PUSHAB  NCB_DESC                                     ;
                                 01  DD 000BE         PUSHL   #1                                           ;
                       00000000G 8F  DD 000C0         PUSHL   #EVL$_OPENMON                                ;
       00000000G  00             04  FB 000C6         CALLS   #4, LIB$SIGNAL                               ;
                                     04 000CD         RET                                                  ;
              22  A6       0000V CF  9E 000CE 4$:     MOVAB   CLOSE_MONITOR, 34(R6)                        ; 2069
      15       0000G  CF         03  E1 000D4         BBC     #3, EVL$GL_LOGMASK, 5$                       ; 2071
                                 7E  D4 000DA         CLRL    -(SP)                                        ; 2073
                  7E        F8   AD  7D 000DC         MOVQ    NCB_DESC, -(SP)                              ;
                                 03  DD 000E0         PUSHL   #3                                           ;
                       00000000G 8F  DD 000E2         PUSHL   #EVL$_LOGOPNM                                ;
       00000000G  00             05  FB 000E8         CALLS   #5, LIB$SIGNAL                               ;
                                     04 000EF 5$:     RET                                                  ; 2075

; Routine Size:  240 bytes,     Routine Base:  $CODE$ + 0A37
```

```
; 1516      2076   1 ROUTINE close_monitor (sink): NOVALUE =
; 1517      2077   1
; 1518      2078   1 !---
; 1519      2079   1 !
; 1520      2080   1 !      Close the monitor process sink.
; 1521      2081   1 !
; 1522      2082   1 ! Inputs:
; 1523      2083   1 !
; 1524      2084   1 !      sink = Address of sink block
; 1525      2085   1 !
; 1526      2086   1 ! Outputs:
; 1527      2087   1 !
; 1528      2088   1 !      None
; 1529      2089   1 !---
; 1530      2090   1
; 1531      2091   2 BEGIN
; 1532      2092   2
; 1533      2093   2 MAP
; 1534      2094   2     sink:       REF BBLOCK;              ! Address of sink block
; 1535      2095   2
; 1536      2096   2 LOCAL
; 1537      2097   2     status,
; 1538      2098   2     ncb_desc:   VECTOR [2];              ! Descriptor of NCB
; 1539      2099   2
; 1540      2100   2 ncb_desc [0] = .sink [sink$b_namelen];  ! Setup descriptor of process name
; 1541      2101   2 ncb_desc [1] = sink [sink$t_name];
; 1542      2102   2
; 1543      2103   2 status = $DASSGN(CHAN = .sink [sink$w_channel]); ! Deassign channel
; 1544      2104   2
; 1545      2105   2 IF NOT .status                         ! If error detected,
; 1546      2106   2 THEN
; 1547      2107   2     SIGNAL(msg(closemon), 1, ncb_desc, .status);
; 1548      2108   2
; 1549      2109   2 sink [sink$l_closertn] = 0;            ! Mark sink closed
; 1550      2110   2
; 1551      2111   1 END;


                                                      .EXTRN  EVL$_CLOSEMON

                                   0004 00000 CLOSE_MONITOR:
                                                      .WORD   Save R2                       ; 2076
               5E            04   C2 00002            SUBL2   #4, SP
               52      04    AC   D0 00005            MOVL    SINK, R2                      ; 2100
               7E      2E    A2   9A 00009            MOVZBL  46(R2), NCB_DESC
         04    AE      2F    A2   9E 0000D            MOVAB   47(R2), NCB_DESC+4            ; 2101
               7E      1E    A2   3C 00012            MOVZWL  30(R2), -(SP)                 ; 2103
   00000000G   00            01   FB 00016            CALLS   #1, SYS$DASSGN
               14            50   E8 0001D            BLBS    STATUS, 1$                    ; 2105
                             50   DD 00020            PUSHL   STATUS                        ; 2107
                       04    AE   9F 00022            PUSHAB  NCB_DESC                      ;
                             01   DD 00025            PUSHL   #1                            ;
                 00000000G   8F   DD 00027            PUSHL   #EVL$_CLOSEMON                ;
   00000000G   00            04   FB 0002D            CALLS   #4, LIB$SIGNAL                ;
                       22    A2   D4 00034 1$:        CLRL    34(R2)                        ; 2109
                                  04 00037            RET                                   ; 2111
```

; Routine Size: 56 bytes,   Routine Base: $CODE$ + 0B27

; R

```
; 1553          2112  1 END
; 1554          2113  0 ELUDOM



                                                                  .EXTRN  LIB$SIGNAL

;                              PSECT SUMMARY
;
;       Name                      Bytes                      Attributes
;
;    $OWN$                          92  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;    $GLOBAL$                        1  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;    $PLIT$                        212  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;    $CODE$                       2911  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      111         1        581         00:01.1
;    _$255$DUA28:[SHRLIB]NET.L32;1          1279        7         0         63         00:00.9
;    _$255$DUA28:[EVL.OBJ]EVCDEF.L32;1       213        0         0         15         00:00.2



;                              COMMAND QUALIFIERS

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

; Size:          2911 code + 305 data bytes
; Run Time:         00:54.1
; Elapsed Time:     01:54.0
; Lines/CPU Min:      2343
; Lexemes/CPU-Min: 27535
; Memory Used:  218 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

0157 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

LPMULT
B32

DRMAST
MAR

ADDRIVER
MAR

TDRIVER
MAR

USSTEST
MAR

GBLSECUFO
MAR

USSDISP
MAR

DOD_ERAPAT
MAR

LBRMAC
MAR

XADRIVER
MAR

LABIOCIN
MAR

DRSLV
MAR

DTE_DF03
MAR

WORKQ
LIS

SCRFT
MAR

EXAMPLES