


```

RRRRRRRR      EEEEEEEEEE      CCCCCCCC      EEEEEEEEEE      I11111      VV      VV      EEEEEEEEEE      RRRRRRRR
RRRRRRRR      EEEEEEEEEE      CCCCCCCC      EEEEEEEEEE      I11111      VV      VV      EEEEEEEEEE      RRRRRRRR
RR      RR      EE      CC      FF      II      VV      VV      EE      RR      RR
RR      RR      EE      CC      FF      II      VV      VV      EE      RR      RR
RR      RR      EE      CC      FF      II      VV      VV      EE      RR      RR
RRRRRRRR      EEEEEEEEEE      CCCCCCCC      EEEEEEEEEE      I11111      VV      VV      EEEEEEEEEE      RRRRRRRR
RRRRRRRR      EEEEEZEEE      CC      RR      RR      RR      RR
RR      RR      FF      CC      FF      II      VV      VV      EEEEEEEEEE      RRRRRRRR
RR      RR      EE      CC      FF      II      VV      VV      EEEEEEEEEE      RR      RR
RR      RR      EE      CC      FF      II      VV      VV      EE      RR      RR
RR      RR      EE      CC      FF      II      VV      VV      EE      RR      RR
RR      RR      EEEEEEEEEE      CCCCCCCC      EEEEEEEEEE      I11111      VV      VV      EEEEEEEEEE      RR      RR
RR      RR      EEEEEEEEEE      CCCCCCCC      EEEEEEEEEE      I11111      VV      VV      EEEEEEEEEE      RR      RR

```

```

LL      I11111      SSSSSSSS
LL      I11111      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
LLLLLLLLLLLL      I11111      SSSSSSSS
LLLLLLLLLLLL      I11111      SSSSSSSS

```

....
....
....
....

```
1 0001 0 MODULE receiver (IDENT = 'V04-000') =
2 0002 1 BEGIN
3 0003 1
4 0004 1
5 0005 1 *****
6 0006 1 *
7 0007 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
8 0008 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
9 0009 1 * ALL RIGHTS RESERVED. *
10 0010 1 *
11 0011 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
12 0012 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
13 0013 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
14 0014 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
15 0015 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
16 0016 1 * TRANSFERRED. *
17 0017 1 *
18 0018 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
19 0019 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
20 0020 1 * CORPORATION. *
21 0021 1 *
22 0022 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
23 0023 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
24 0024 1 *
25 0025 1 *
26 0026 1 *****
27 0027 1
28 0028 1 ++
29 0029 1 FACILITY: DECnet V2.0 event logger
30 0030 1
31 0031 1 ABSTRACT:
32 0032 1
33 0033 1 This module contains the routines which receive events from
34 0034 1 both the local node or other nodes and place the events into
35 0035 1 the appropriate sink (console, file or monitor process).
36 0036 1
37 0037 1 ENVIRONMENT:
38 0038 1
39 0039 1 VAX/VM, operating system. unprivileged user mode,
40 0040 1
41 0041 1 AUTHOR: Tim Halvorsen, June 1980
42 0042 1
43 0043 1 Modified by:
44 0044 1
45 0045 1 V004 MKP0001 Kathy Perko 24-June-1984
46 0046 1 Increase size of OPCOM message buffer to prevent truncation
47 0047 1 of events logged to OPCOM.
48 0048 1
49 0049 1 V003 TMH0003 Tim Halvorsen 20-Jul-1983
50 0050 1 Pass back our version number in connect accept to remote
51 0051 1 event transmitter to conform to the architecture, and allow
52 0052 1 PLUTOs to send us events (they didn't like our lack of
53 0053 1 version number).
54 0054 1 Always send events to the OPCOM facility as well as
55 0055 1 sending them to the monitor process (if any); rather
56 0056 1 than doing one or the other.
57 0057 1
```

RECEIVER
V04-000

C 14
16-Sep-1984 01:37:57
14-Sep-1984 12:28:54

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EVL.SRC]RECEIVER.B32;1 Page 2 (1)

```

: 58      0058 1 |      V002  TMH0002      Tim Halvorsen  02-May-1983
: 59      0059 1 |      Fix bug which left links permanently accessed but in
: 60      0060 1 |      the closed state; the links weren't being deaccessed
: 61      0061 1 |      if an error was encountered issuing a link receive.
: 62      0062 1 |
: 63      0063 1 |      V001  TMH0001      Tim Halvorsen  03-Jun-1982
: 64      0064 1 |      Convert to use new NETACP control QIO interface.
: 65      0065 1 |      --
: 66      0066 1 |
: 67      0067 1 |
: 68      0068 1 |      Include files
: 69      0069 1 |
: 70      0070 1 |
: 71      0071 1 | LIBRARY 'SYSS$LIBRARY:STARLET';      ! VAX/VMS common definitions
: 72      0072 1 |
: 73      0073 1 | REQUIRE 'SYSS$LIBRARY:UTILDEF';      ! Common VMS definitions
: 74      0249 1 |
: 75      0250 1 | LIBRARY 'SHRLIBS:NET';              ! Network control QIO definitions
: 76      0251 1 |              ! (NETACP database symbols)
: 77      0252 1 |
: 78      0253 1 | LIBRARY 'LIBS:EVCDEF';              ! Event data definitions
: 79      0254 1 |
: 80      0255 1 | REQUIRE 'LIBS:EVLDEF';              ! Event logger definitions

```

RE
VO

```

82 0664 1  |
83 0665 1  | Table of contents
84 0666 1  |
85 0667 1  |
86 0668 1  | FORWARD ROUTINE
87 0669 1  |   evl$receive:      NOVALUE,  | Receiver initialization
88 0670 1  |   update_sinkdata: NOVALUE,  | Update our version of sink data
89 0671 1  |   delete_sink:     NOVALUE,  | Delete a sink
90 0672 1  |   wait_for_interrupt: NOVALUE, | Wait for network interrupt
91 0673 1  |   net_interrupt:   NOVALUE,  | Network interrupt AST routine
92 0674 1  |   open_event_link: NOVALUE,  | Open incoming event logical link
93 0675 1  |   close_event_link: NOVALUE, | Close event logical link
94 0676 1  |   shutdown:        NOVALUE,  | Shut down receiver gracefully
95 0677 1  |   wait_for_event:  NOVALUE,  | Wait for an incoming event record
96 0678 1  |   event_received:  NOVALUE,  | Accept incoming event record
97 0679 1  |   evl$queue_event: NOVALUE,  | Queue an event record to sinks
98 0680 1  |   output_events:   NOVALUE,  | Output all events for a given sink
99 0681 1  |   output_console:  NOVALUE,  | Output event to console sink
100 0682 1  |   write_line:      NOVALUE,  | Write line to console sink device
101 0683 1  |   output_file:     NOVALUE,  | Output event to file sink
102 0684 1  |   open_file:       NOVALUE,  | Open file sink
103 0685 1  |   close_file:      NOVALUE,  | Close file sink
104 0686 1  |   output_opcom:    NOVALUE,  | Output event to OPCOM terminals
105 0687 1  |   output_monitor:  NOVALUE,  | Output event to process sink
106 0688 1  |   save_lines:      NOVALUE,  | Append output lines into buffer
107 0689 1  |   open_monitor:    NOVALUE,  | Open monitor process sink
108 0690 1  |   close_monitor:   NOVALUE,  | Close monitor process sink
109 0691 1  |
110 0692 1  |
111 0693 1  | BUILTIN functions
112 0694 1  |
113 0695 1  |
114 0696 1  | BUILTIN
115 0697 1  |   INSQUE,          | INSQUE instruction
116 0698 1  |   REMQUE:          | REMQUE instruction
117 0699 1  |
118 0700 1  |
119 0701 1  | Define macro for message reporting
120 0702 1  |
121 0703 1  |
122 0704 1  | MACRO
123 M 0705 1  |   msg(ident) =
124 M 0706 1  |     BEGIN
125 M 0707 1  |       %IF NOT %DECLARED(%NAME('evl$ ',ident))
126 M 0708 1  |         %THEN EXTERNAL LITERAL %NAME('evl$_',ident); %FI
127 M 0709 1  |       %NAME('evl$_',ident)
128 0710 1  |     END%;
129 0711 1  |
130 0712 1  |
131 0713 1  | Literals
132 0714 1  |
133 0715 1  |
134 0716 1  | LITERAL
135 0717 1  |   object_number = 26,  | EVL object number
136 0718 1  |   max_queued_events = 10, | Maximum events on sink queue before
137 0719 1  |                           | incoming events are blocked and the
138 0720 1  |                           | sink queue is emptied

```

```

: 139      0721      1      mbx_maxmsg = 64;                ! Maximum size of mailbox message
: 140      0722      1
: 141      0723      1      !
: 142      0724      1      !: OWN storage
: 143      0725      1
: 144      0726      1
: 145      0727      1      OWN
: 146      0728      1      net_channel:      WORD,                ! Logical link channel number
: 147      0729      1      mbx_channel:      WORD,                ! Mailbox channel number
: 148      0730      1      mbx_message:      VECTOR [mbx_maxmsg, BYTE], ! Mailbox input buffer
: 149      0731      1      iosb:      BBLOCK [8],                ! I/O status block for MBX
: 150      0732      1      sink_header:      VECTOR [2],                ! Listhead for sink descriptors
: 151      0733      1      iec_header:      VECTOR [2];                ! Listhead for incoming links
: 152      0734      1
: 153      0735      1      GLOBAL
: 154      0736      1      evl$b_rcvdone:      BYTE INITIAL(true);    ! True if receiver inactive
: 155      0737      1
: 156      0738      1      !
: 157      0739      1      !: External storage
: 158      0740      1
: 159      0741      1
: 160      0742      1      EXTERNAL
: 161      0743      1      evl$gl_logmask:      BBLOCK,                ! Logging bit mask
: 162      0744      1      evl$gt_localnode:      ! Local node name string
: 163      0745      1
: 164      0746      1      !
: 165      0747      1      !: External routines
: 166      0748      1
: 167      0749      1
: 168      0750      1      EXTERNAL ROUTINE
: 169      0751      1      format_event:      NOVALUE,                ! Format an event record
: 170      0752      1      evl$allocdbk,                ! Allocate data storage block
: 171      0753      1      evl$deallocdbk,                ! Deallocate data storage block
: 172      0754      1      evl$unjulian,                ! Convert julian half-day time
: 173      0755      1      evl$sprintlog,                ! Print hex bytes to log
: 174      0756      1      evl$netshow,                ! Get information from NETACP
: 175      0757      1      wkq$add_work_item,                ! Add work to work queue
: 176      0758      1      lib$asn_wth_mbx: ADDRESSING MODE(GENERAL), ! Assign with assoc. mailbox
: 177      0759      1      lib$put_output: ADDRESSING MODE(GENERAL), ! Write to SYSS$OUTPUT
: 178      0760      1      lib$get_vm: ADDRESSING MODE(GENERAL),    ! Allocate storage
: 179      0761      1      lib$free_vm: ADDRESSING_MODE(GENERAL);    ! Deallocate storage

```

```

181 0762 1 GLOBAL ROUTINE evl$receive: NOVALUE =
182 0763 1
183 0764 1 ---
184 0765 1
185 0766 1 This routine is called to initialize the event
186 0767 1 receiver and setup work to do asynchronously
187 0768 1 via the work queue. A request for sink data is
188 0769 1 issued to NETACP and nothing is done until the
189 0770 1 request is satisfied. When sink data is obtained
190 0771 1 (or when there is any), then the sink list is
191 0772 1 updated and processing of events begins.
192 0773 1
193 0774 1 Inputs:
194 0775 1
195 0776 1 None
196 0777 1
197 0778 1 Outputs:
198 0779 1
199 0780 1 None
200 0781 1 ---
201 0782 1
202 0783 2 BEGIN
203 0784 2
204 0785 2 LOCAL
205 0786 2 nfb: BBLOCK [5], ! Network function block for DECLOBJ
206 0787 2 nfb_desc: VECTOR [2], ! Descriptor of NFB
207 0788 2 mbx_modes: VECTOR [2], ! Enables mailbox message types
208 0789 2 status;
209 0790 2
210 0791 2 sink_header [0] = sink_header; ! Initialize sink listhead
211 0792 2 sink_header [1] = sink_header;
212 0793 2
213 0794 2 iec_header [0] = iec_header; ! Initialize incoming channel listhead
214 0795 2 iec_header [1] = iec_header;
215 0796 2
216 0797 2 status = LIB$ASN_WTH_MBX(%ASCID '_NET:', ! Assign channel to NETACP
217 0798 2 0,0, ! mailbox MAXMSG, BUFQUO (ignored)
218 0799 2 net_channel, ! Channel to NETACP
219 0800 2 mbx_channel); ! Channel to mailbox
220 0801 2
221 0802 2 IF NOT .status ! If error assigning channel,
222 0803 2 THEN
223 0804 2 BEGIN
224 0805 2 SIGNAL(msg(netasn), 0, .status); ! then signal the error
225 0806 2 RETURN;
226 0807 2 END;
227 0808 2
228 0809 2 nfb [0,0,8,0] = NFB$C_DECLOBJ; ! Set function to 'Declare object'
229 0810 2 nfb [1,0,32,0] = object_number; ! Object number for event logger
230 0811 2
231 0812 2 nfb_desc [0] = 5; ! Setup descriptor of NFB
232 0813 2 nfb_desc [1] = nfb;
233 0814 2
234 P 0815 2 status = $QIOW(FUNC = IOS$ACPCONTROL, ! Issue read on mailbox
235 P 0816 2 CHAN = .net_channel,
236 P 0817 2 EFN = evl$c_synch_efn,
237 P 0818 2 IOSB = iosb,

```


0004C SINK_HEADER:
 .BKLB 8
00054 IEC_HEADER:
 .BKLB 8
 .PSECT \$GLOBALS,NOEXE,2

01 00000 EVL\$B_RCVDONE::
 .BYTE 1

.EXTRN EVL\$GL_LOGMASK, EVL\$GT_LOCALNODE
.EXTRN FORMAT_EVENT, EVL\$ALLOCDKB
.EXTRN EVL\$DEALLOCDKB, EVL\$SUNJULIAN
.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 SYSS\$QIOW, SYSSEXIT

.PSECT \$CODES,NOWRT,2

			007C 00000	.ENTRY	EVL\$RECEIVE, Save R2,R3,R4,R5,R6	: 0762
	56	00000000G	00 9E 00002	MOVAB	SYSS\$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	:
	00		05 FB 0003F	CALLS	#5, LIB\$ASN_WTH_MBX	:
	32		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		16 90 00056	MOVB	#22, NFB	: 0809
	11		1A D0 0005A	MOVL	#26, NFB+1	: 0810
	08		05 D0 0005E	MOVL	#5, NFB_DESC	: 0812
	0C		AE 9E 00062	MOVAB	NFB, NFB_DESC+4	: 0813
		10	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	:

RE
VO
:
:
:

66	0C	FB	0007C	CALLS	#12, SYSSQIOW	
52	50	DO	0007F	MOVL	R0, STATUS	
03	52	F9	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, SYSSEXIT
	09	52	E8	00096	3\$:	BLBS STATUS, 4\$
		52	DD	00099	PUSHL	STATUS
		7E	D4	0009B	CLRL	-(SP)
		54	DD	0009D	PUSHL	R4
65	03	FB	0009F	CALLS	#3, LIBSSIGNAL	
	04	AE	000A2	4\$:	CLRL MBXMODES	0834
		01	CE	000A4	MNEGL	#1, MBXMODES+4
		7E	7C	000A8	CLRQ	-(SP)
		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
		01	DD	000BB	PUSHL	#1
		0C	FB	000BD	CALLS	#12, SYSSQIOW
	66	50	DO	000C0	MOVL	R0, STATUS
	52	52	F9	000C3	BLBC	STATUS, 5\$
	06	63	3C	000C6	MOVZWL	IOSB, STATUS
	52	52	E8	000C9	BLBS	STATUS, 6\$
	09	52	DD	000CC	5\$:	PUSHL STATUS
		7E	D4	000CE	CLRL	-(SP)
		54	DD	0G0D0	PUSHL	R4
	65	03	FB	000D2	CALLS	#3, LIBSSIGNAL
0000V	CF	00	FB	000D5	6\$:	CALLS #0, UPDATE_SINKDATA
0000V	CF	00	FB	000DA	CALLS	#0, WAIT_FOR_INTERRUPT
		04	000DF	RET		0851
						0853
						0855

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

```

276 0856 1 ROUTINE update_sinkdata: NOVALUE =
277 0857 1
278 0858 1 ---
279 0859 1
280 0860 1 This routine is called to request an updated description
281 0861 1 of the sink types from NETACP.
282 0862 1
283 0863 1 The format of the sink information in NETACP is:
284 0864 1
285 0865 1 0) Sink type (console, file or monitor process)
286 0866 1 4) State (on, off or hold)
287 0867 1 8) Sink name string (counted string)
288 0868 1
289 0869 1 Inputs:
290 0870 1
291 0871 1 None
292 0872 1
293 0873 1 Outputs:
294 0874 1
295 0875 1 sink_header = Address of sink listhead
296 0876 1 evl$b_rcvdone = True if list empty (receiver inactive), else false
297 0877 1
298 0878 1 ---
299 0879 1
300 0880 2 BEGIN
301 0881 2
302 0882 2 LOCAL
303 0883 2 status,
304 0884 2 buffer: BBLOCK [256], ! Buffer for sink information
305 0885 2 bufdesc: VECTOR [2], ! Descriptor of above buffer
306 0886 2 position: VECTOR [nfb$c_ctx_size, BYTE], ! Buffer for current position
307 0887 2 sink: REF BBLOCK; ! Address of current sink entry
308 0888 2
309 0889 2 sink = .sink_header; ! Start at first sink block
310 0890 2
311 0891 2 WHILE .sink NEQ sink_header ! Until end of linked list,
312 0892 2 DO
313 0893 2 BEGIN
314 0894 2 sink [sink$v_delete] = true; ! Mark for possible deletion
315 0895 2 sink = .sink [sink$l_link]; ! and link to next one
316 0896 2 END;
317 0897 2
318 0898 2 bufdesc [0] = 256; ! Setup descriptor
319 0899 2 bufdesc [1] = buffer;
320 0900 2
321 0901 2 position <0,16> = 0; ! Start at first ESI record
322 0902 2
323 0903 2 WHILE evl$netshow(
324 0904 2 nfb$c_db_esi, ! Event sink information
325 0905 2 nfb$c_wildcard,0, ! Search all records
326 0906 2 position, ! Get next record; update position
327 0907 2 3, ! # fields for each item; fields are:
328 0908 2 UPLIT( nfb$c_esi_snk, ! Sink type (longword)
329 0909 2 nfb$c_esi_sta, ! Sink state (longword)
330 0910 2 nfb$c_esi_lna), ! Sink name (string)
331 0911 2 bufdesc) ! Return buffer descriptor
332 0912 2 DO

```

```

333 0913 3 BEGIN
334 0914 3 IF .evl$gl_logmask [elg$vbupdat] ! If we are logging database updates,
335 0915 3 THEN
336 0916 3 BEGIN
337 0917 3 SIGNAL(msg(logdbur), 1, 0); ! then signal update occurred
338 0918 3 SIGNAL(msg(dbcrcv), 4,
339 0919 3 CASE .buffer [0,0,32,0] FROM sink$c_console TO sink$c_monitor
340 0920 3 OF
341 0921 3 SET
342 0922 3 [sink$c_console]: UPLIT BYTE(%ASCIC 'console');
343 0923 3 [sink$c_file]: UPLIT BYTE(%ASCIC 'file');
344 0924 3 [sink$c_monitor]: UPLIT BYTE(%ASCIC 'monitor_process');
345 0925 3 [OUTRANGE]: UPLIT BYTE(%ASCIC 'unknown');
346 0926 3 TES,
347 0927 3 .buffer [8,0,16,0], buffer [10,0,0,0],
348 0928 3 CASE .buffer [4,0,32,0] FROM sink$c_on TO sink$c_hold
349 0929 3 OF
350 0930 3 SET
351 0931 3 [sink$c_on]: UPLIT BYTE(%ASCIC 'ON');
352 0932 3 [sink$c_off]: UPLIT BYTE(%ASCIC 'OFF');
353 0933 3 [sink$c_hold]: UPLIT BYTE(%ASCIC 'HOLD');
354 0934 3 [OUTRANGE]: UPLIT BYTE(%ASCIC 'UNKNOWN');
355 0935 3 TES);
356 0936 3 END;
357 0937 3
358 0938 3 sink = .sink_header; ! Start at first entry in list
359 0939 3
360 0940 3 IF
361 0941 3 WHILE .sink NEQ sink_header ! Until end of list
362 0942 3 DO
363 0943 3 BEGIN
364 0944 3 IF .sink [sink$b_type] EQL .buffer [0,0,32,0] ! If sink type matches
365 0945 3 THEN
366 0946 3 BEGIN
367 0947 3 LOCAL previous_state;
368 0948 3 previous_state = .sink [sink$b_state]; ! Save previous state
369 0949 3 sink [sink$b_state] = .buffer [4,0,32,0]; ! Store new state
370 0950 3 sink [sink$b_namelen] = .buffer [8,0,16,0]; ! Store new name
371 0951 3 CHSMOVE(.sink [sink$b_namelen], buffer [10,0,0,0], sink [sink$t_name]);
372 0952 3 sink [sink$v_error] = false; ! Try operations on sink again
373 0953 3 IF .sink [sink$b_state] NEQ sink$c_off ! If sink being turned on,
374 0954 3 OR .previous_state EQL sink$c_off ! or if sink in the process
375 0955 3 ! of being deleted,
376 0956 3 THEN
377 0957 3 sink [sink$v_delete] = false; ! Mark this sink be retained
378 0958 3 EXITLOOP false;
379 0959 3 END;
380 0960 3 sink = .sink [sink$l_link]; ! Skip to next in chain
381 0961 3 END
382 0962 3 ! If not found in list,
383 0963 3 AND .buffer [4,0,32,0] NEQ sink$c_off ! and not in 'off' state,
384 0964 3 THEN
385 0965 3 BEGIN ! Then add the sink
386 0966 3 LOCAL
387 0967 3 length;
388 0968 3 length = sink$c_length; ! Length of new sink entry
389 0969 3 signal_if_error(LIB$GET_VM(length,sink)); ! Allocate new entry

```

```

390      CH$FILL(0,sink$c_length,.sink);           ! Zero the entry
391      sink [sink$b_type] = .buffer [0,0,32,0];   ! Store sink type
392      sink [sink$b_state] = .buffer [4,0,32,0];  ! Store new state
393      sink [sink$b_namelen] = .buffer [8,0,16,0]; ! Store new name
394      CH$MOVE(.sink [sink$b_namelen], buffer [10,0,0,0], sink [sink$t_name]);
395      sink [sink$l_evtfl] = sink [sink$l_evtfl]; ! Init event queue listhead
396      sink [sink$l_evtbl] = sink [sink$l_evtfl];
397      INSQUE(.sink, .sink_header [1]);           ! Insert into queue
398      evl$b_rcvdone = false;                       ! Mark receiver active
399      END;
400
401      IF .sink [sink$b_state] EQL sink$c_on       ! If sink just turned on,
402      THEN
403      wkq$add_work_item(output_events,.sink); ! then flush any 'held' events
404      END;
405
406      !-----
407      ! If any sinks were turned off, we flush any events queued
408      ! for that sink and then delete the sink. These two functions
409      ! are queued on the work queue to ensure that any asynchronous
410      ! activity on that sink completes before the delete is done.
411      !-----
412
413      sink = .sink_header;                         ! Start at first sink again
414
415      WHILE .sink NEQ sink_header                 ! For each sink block
416      DO
417      BEGIN
418      IF .sink [sink$v_delete]                   ! If sink was turned off or omitted,
419      THEN
420      BEGIN
421      sink [sink$b_state] = sink$c_off;          ! Set state off if sink omitted
422      wkq$add_work_item(output_events,.sink); ! then flush the sink events
423      wkq$add_work_item(delete_sink,.sink);    ! and delete the sink
424      END;
425      sink = .sink [sink$l_link];                ! Skip to next in list
426      END;
427
428      END;

```

										.PSECT \$PLITS,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\							
73	65	63	6F	72	70	20	72	6F	74	69	6E	6F	6D	0F	00024	P.AAE:	.ASCII	<4>\file\
															00029	P.AAF:	.ASCII	<15>\monitor process\
															00038			
															00039	P.AAG:	.ASCII	<7>\unknown\
															00041	P.AAH:	.ASCII	<2>\ON\
															00044	P.AAI:	.ASCII	<3>\OFF\
															00048	P.AAJ:	.ASCII	<4>\HOLD\
															0004D	P.AAK:	.ASCII	<7>\UNKNOWN\
										.EXTRN EVL\$_LOGDBUR, EVL\$_DBCRCV								

.PSECT \$CODE\$,NOWRT,2

OFFC 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
	50		6E	DD	0001F	1\$:	MOVL
	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)
		6E	60	DD	0002E	MOVL	(R0), SINK
			EC	11	00031	BRB	1\$
	48	AE	8F	3C	00033	2\$:	MOVZWL
	4C	AE	50	AE	9E	00039	MOVAB
			08	AE	B4	0003E	CLRW
			48	AE	9F	00041	3\$:
			58	DD	00044	PUSHAB	BUFDESC
			03	DD	00046	PUSHL	R8
			14	AE	9F	00048	PUSHL
	7E		01	7D	0004B	PUSHAB	POSITION
			07	DD	0004E	MOVQ	#1, -(SP)
	0000G	CF	07	FB	00050	PUSHL	#7
			50	EB	00055	CALLS	#7, EVL\$NETSHOW
			011A	31	00058	BLBS	R0, 4\$
	64	0000G	CF	E9	0005B	4\$:	BRW
	7E		01	7D	00060	BLBC	EVL\$GL LOGMASK, 15\$
			8F	DD	00063	MOVQ	#1, -(SP)
	6A	00000000G	03	FB	00069	PUSHL	#EVL\$ LOGDBUR
	00		AE	CF	0006C	CALLS	#3, LIB\$SIGNAL
02	0018	0012	54	000C	00071	5\$:	CASEL
							BUFFER+4, #0, #2
							.WORD
							6\$-5\$,-
							7\$-5\$,-
							8\$-5\$
	50	3D	A8	9E	00077	MOVAB	P.AAK, R0
			10	11	0007B	BRB	9\$
	50	31	A8	9E	0007D	6\$:	MOVAB
			0A	11	00081	BRB	9\$
	50	34	A8	9E	00083	7\$:	MOVAB
			04	11	00087	BRB	9\$
	50	38	A8	9E	00089	8\$:	MOVAB
			50	DD	0008D	9\$:	PUSHL
			5E	AE	9F	0008F	PUSHAB
	7E		60	AE	3C	00092	MOVZWL
02	0018	0012	5C	AE	CF	00096	CASEL
			000C		0009B	10\$:	BUFFER, #1, #2
							.WORD
							11\$-10\$,-
							12\$-10\$,-
							13\$-10\$
	50	29	A8	9E	000A1	MOVAB	P.AAG, R0
			10	11	000A5	BRB	14\$
	50	0C	A8	9E	000A7	11\$:	MOVAB
			0A	11	000AB	BRB	14\$
	50	14	A8	9E	000AD	12\$:	MOVAB
			04	11	000B1	BRB	14\$

				50	19	A8	9E	000B3	13\$:	MOVAB	P.AAF, R0	0924	
						50	DD	000B7	14\$:	PUSHL	R0	0927	
						04	DD	000B9		PUSHL	#4	0938	
					00000000G	8F	DD	000BB		PUSHL	#EVL\$ DBCRCV	0941	
				6A		06	FB	000C1		CALLS	#6, LIB\$SIGNAL		
				6E		69	DO	000C4	15\$:	MOVL	SINK_HEADER, SINK		
				56		6E	DO	000C7	16\$:	MOVL	SINK, R6		
				50		69	9E	000CA		MOVAB	SINK_HEADER, R0		
				50		56	D1	000CD		CMPL	R6, R0		
						3B	13	000D0		BEQL	19\$		
50	AE			08		00	ED	000D2		CMPL	#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	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	DO	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
							AE	9F	00115		PUSHAB	LENGTH	
					00000000G	00	02	FB	00118		CALLS	#2, LIB\$GET_VM	
						52	50	DO	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	DO	0012B	20\$:	MOVL	SINK, R6	0970	
012E	8F			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	A6	2E	9A	00145		MOVZBL	46(R6), R0	0974	
			2F	A6	5A	AE	50	28	00149	MOVC3	R0, BUFFER+10, 47(R6)		
					0C	A6	9E	0014F		MOVAB	12(R6), 12(R6)	0975	
				10	A6	0C	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	DO	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		
				68		02	FB	0016F		CALLS	#2, WKQ\$ADD_WORK_ITEM		
						FEC	31	00172	22\$:	BRW	3\$	0903	
				6E		69	DO	00175	23\$:	MOVL	SINK_HEADER, SINK	0993	
				52		6E	DO	00178	24\$:	MOVL	SINK, R2	0995	
				50		69	9E	00178		MOVAB	SINK_HEADER, R0		

RECEIVER
V04-000

B 15
16-Sep-1984 01:37:57
14-Sep-1984 12:28:54

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EVL.SRC]RECEIVER.B32;1 Page 14
(4)

50		52	D1	0017E	C MPL	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, WKQ\$ADD_WORK_ITEM	:
	0000V	52	DD	00194	PUSHL	R2	1003
6B		02	FB	0019A	PUSHAB	DELETE_SINK	:
6E		62	D0	0019D 25\$:	CALLS	#2, WKQ\$ADD_WORK_ITEM	1005
		D6	11	001A0	MOVL	(R2), SINK	0995
		04	001A2 26\$:	BRB	24\$:	1008
					RET		:

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


```

: 430 1009 1 ROUTINE delete_sink (sink): NOVALUE =
: 431 1010 1
: 432 1011 1 |---
: 433 1012 1
: 434 1013 1 |       Delete a given sink
: 435 1014 1
: 436 1015 1 |   Inputs:
: 437 1016 1
: 438 1017 1 |       sink = Address of sink block
: 439 1018 1
: 440 1019 1 |   Outputs:
: 441 1020 1
: 442 1021 1 |       None
: 443 1022 1 |---
: 444 1023 1
: 445 1024 2 BEGIN
: 446 1025 2
: 447 1026 2 MAP
: 448 1027 2   sink:      REF BBLOCK;      ! Address of sink block
: 449 1028 2
: 450 1029 2 LOCAL
: 451 1030 2   length;
: 452 1031 2
: 453 1032 2 IF .sink [sink$_closertn] NEQ 0      ! If there is a close routine,
: 454 1033 2 THEN
: 455 1034 2   (.sink [sink$_closertn])(.sink); ! Close the sink
: 456 1035 2
: 457 1036 2 REMQUE(.sink,length);              ! Remove from linked list
: 458 1037 2
: 459 1038 2 length = sink$_length;
: 460 1039 2 LIB$FREE_VM(length,sink);          ! Deallocate sink block
: 461 1040 2
: 462 1041 3 evl$b_rcvdone = (.sink_header [0] EQL sink_header) ! Flag if we are active
: 463 1042 2   AND (.iec_header [0] EQL iec_header);
: 464 1043 2
: 465 1044 1 END;

```

```

                                001C 0000 DELETE_SINK:
                                .WORD   Save R2,R3,R4
                                54      0000' CF 9E 00002   MOVAB   SINK_HEADER, R4      : 1009
                                5E      04   C2 00007   SUBL2  #4, SP
                                52      04   AC D0 0000A   MOVL   SINK, R2             : 1032
                                22      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      6E   62 0F 00019 1$: REMQUE (R2), LENGTH      : 1036
                                6E      012E 8F 3C 0001C   MOVZWL #302, LENGTH      : 1038
                                04      04   AC 9F 00021   PUSHAB SINK                : 1039
                                04      04   AE 9F 00024   PUSHAB LENGTH
                                00000000G 00 02 FB 00027   CALLS #2, LIB$FREE_VM
                                50      64 9E 0002E   MOVAB  SINK_HEADER, -R0
                                52      D4 00031   CLRL  R2                   : 1041

```

RECEIVER
V04-000

D 15
16-Sep-1984 01:37:57
14-Sep-1984 12:28:54

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EVL.SRC]RECEIVER.B32;1 Page 16
(5)

RE
VO

		50		64	D1	00033		C MPL	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		C MPL	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 = $QIO(FUNC = IOS_READVBLK,      ! Issue read on mailbox
490 P 1068 2     CHAN = .mbx_channel,
491 P 1069 2     EFN = evl$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 SYSSQIO, EVL\$_READEV

```

0000 0000 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
0000000G 00 0C FB 00021 CALLS #12, SYSSQIO : 1075
11 50 E8 00028 BLBS STATUS, 1$ : 1077
50 DD 0002B PUSHL STATUS
7E D4 0002D CLRL -(SP)
0000000G 8F DD 0002F PUSHL #EVL$_READEV

```

RECEIVER
V04-000

F 15
16-Sep-1984 01:37:57
14-Sep-1984 12:28:54

VAX-11 BLISS-32 V4.0-742
DISK\$VMSMASTER:[EVL.SRC]RECEIVER.B32;1 Page 18 (6)

00000000G 00

03 FB 00035 CALLS #3, LIB\$SIGNAL
04 0003C 1\$: RET

:
: 1079

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

RE
VO

:

```

503 1080 1 ROUTINE net_interrupt: NOVALUE =
504 1081 1
505 1082 1 ---
506 1083 1
507 1084 1 This AST routine is called when the outstanding QIO
508 1085 1 on the associated mailbox completes. If the interrupt
509 1086 1 indicates a connect is pending, then the acceptance
510 1087 1 routine is added to the work queue.
511 1088 1
512 1089 1 Inputs:
513 1090 1
514 1091 1     mbx_message = Mailbox message
515 1092 1
516 1093 1 Outputs:
517 1094 1
518 1095 1     None
519 1096 1 ---
520 1097 1
521 1098 2 BEGIN
522 1099 2
523 1100 2 SELECTONEU .mbx_message [0]           ! Select based on message type
524 1101 2 OF
525 1102 2     SET
526 1103 2
527 1104 2 [msg$_netshut]:                   ! Network shutting down
528 1105 2
529 1106 2     BEGIN
530 1107 2     wkq$add_work_item(shutdown);      ! Shut down receiver gracefully
531 1108 2     RETURN;                          ! Do not re-issue mailbox read
532 1109 2     END;
533 1110 2
534 1111 2 [msg$_connect]:                       ! Incoming connect request
535 1112 2
536 1113 2     BEGIN
537 1114 2     LOCAL
538 1115 2         ptr,len,
539 1116 2         iec: REF BBLOCK;              ! Incoming event channel block
540 1117 2     len = iec$length;
541 1118 2     LIB$GET_VM(len,iec);               ! Allocate incoming channel block
542 1119 2     CH$FILL(0,iec$length,.iec);       ! Zero the block
543 1120 2     iec [iec$w_size] = iec$length;    ! Set length of block
544 1121 2     ptr = 5 + .mbx_message [4];       ! Get index of start of ascic data
545 1122 2     iec [iec$b_ncb(len)] = .mbx_message [.ptr]; ! Set length of NCB
546 1123 2     CH$COPY(.mbx_message [.ptr], mbx_message [.ptr+1],
547 1124 2         0, iec$maxncb(len, iec [iec$ncb]));
548 1125 2     INSQUE(.iec, .iec header [1]);    ! Insert into list
549 1126 2     wkq$add_work_item(open_event_link,.iec); ! Queue the connect accept
550 1127 2     evl$b_rcvdone = false;           ! Mark receiver active
551 1128 2     END;
552 1129 2
553 1130 2 [msg$_evtrcvchg]:                   ! If event receiver database change
554 1131 2
555 1132 2     wkq$add_work_item(update_sinkdata); ! then update sink information
556 1133 2
557 1134 2 TES;
558 1135 2
559 1136 2 wait_for_interrupt();               ! Issue another read on mailbox

```

: 560 1137 2
: 561 1138 1 END;

			01FC 00000 NET_INTERRUPT:								
			58	0000G	CF	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8		1080
			57	0000'	CF	9E	00007	MOVAB	WKQ\$ADD_WORK_ITEM, R8		
			5E		08	C2	0000C	MOVAB	MBX_MESSAGE, R7		
			50		67	9A	0000F	SUBL2	#8, SP		1100
			3B		50	91	00012	MOVZBL	MBX_MESSAGE, R0		1104
					08	12	00015	CMPB	R0, #59		
				0000V	CF	9F	00017	BNEQ	1\$		
			68		01	FB	0001B	PUSHAB	SHUTDOWN		1107
						04	0001E	CALLS	#1, WKQ\$ADD_WORK_ITEM		
			32		50	91	0001F	RET			1106
					51	12	00022	CMPB	R0, #50		1111
		04	AE	014F	8F	3C	00024	BNEQ	2\$		
					5E	DD	0002A	MOVZWL	#335, LEN		1117
				08	AE	9F	0002C	PUSHL	SP		1118
		00000000G	00		02	FB	0002F	PUSHAB	LEN		
			56		6E	DD	00036	CALLS	#2, LIB\$GET_VM		
014F	8F	00	6E		00	2C	00039	MOVL	IEC, R6		1119
					66		00040	MOVCS	#0, (SP), #0, #335, (R6)		
			08	A6	014F	8F	00041	MOVW	#335, 8(R6)		1120
			50		04	A7	00047	MOVZBL	MBX_MESSAGE+4, PTR		1121
			50			05	0004B	ADDL2	#5, PTR		
			14	A6	6740	90	0004E	MOVW	MBX_MESSAGE[PTR], 20(R6)		1122
			51		6740	9A	00053	MOVZBL	MBX_MESSAGE[PTR], R1		1123
0040	8F	00	01	A740	51	2C	00057	MOVCS	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, WKQ\$ADD_WORK_ITEM		
				0000'	CF	94	0006F	CLRB	EVL\$B_RCVDOFE		1127
					0C	11	00073	BRB	3\$		1100
			3F		50	91	00075	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	CALLS	#0, WAIT_FOR_INTERRUPT		1136
					04		00086	RET			1138

: Routine Size: 135 bytes, Routine Base: \$CODE\$ + 0312

```

563 1139 1 ROUTINE open_event_link (iec): NOVALUE =
564 1140 1
565 1141 1 |---
566 1142 1 |
567 1143 1 |       Open the logical link for incoming event records.
568 1144 1 |
569 1145 1 |       Inputs:
570 1146 1 |
571 1147 1 |       iec = Address of incoming event channel block
572 1148 1 |
573 1149 1 |       Outputs:
574 1150 1 |
575 1151 1 |       routine = True if link established, false if not
576 1152 1 |---
577 1153 1
578 1154 2 BEGIN
579 1155 2
580 1156 2 MAP
581 1157 2     iec:          REF BBLOCK;          ! Address of event channel block
582 1158 2
583 1159 2 LOCAL
584 1160 2     status,
585 1161 2     ptr,
586 1162 2     ncb_desc:  VECTOR [2];          ! Descriptor of NCB
587 1163 2
588 1164 2 |
589 1165 2 |       Setup NCB for connect accept
590 1166 2 |
591 1167 2
592 1168 2 ncb_desc [0] = .iec [iec$b_ncblen];    ! Get length of requestor ncb
593 1169 2 ncb_desc [1] = iec [iec$t_ncb];      ! and address of ncb
594 1170 2
595 1171 2 |
596 1172 2 |       Get copy of NCB up to slash to enable its use in error reporting
597 1173 2 |
598 1174 2
599 1175 2 ptr = CH$FIND_CH(.iec [iec$b_ncblen], iec [iec$t_ncb], '/');
600 1176 2
601 1177 2 IF NOT CH$FAIL(.ptr)                  ! If ending slash found,
602 1178 2 THEN
603 1179 2     iec [iec$b_ncblen] = .ptr - iec [iec$t_ncb]; ! then truncate rest of junk
604 1180 2
605 1181 2 |
606 1182 2 |       Accept the connect request always since we try to support ANY version
607 1183 2 |       of remote event transmitter. Pass back our version number in the
608 1184 2 |       "connect accept" optional data.
609 1185 2 |
610 1186 2
611 1187 2 CH$MOVE(4, UPLIT BYTE(3, 4,0,0), .ptr+3); ! Pass our version # in optional data
612 1188 2
613 1189 2 P status = $ASSIGN(DEVNAM = %ASCII ' _NET:', ! Get channel for incoming link
614 1190 2     CHAN = iec [iec$w_chan]);
615 1191 2
616 1192 2 IF NOT .status                        ! If error assigning channel,
617 1193 2 THEN
618 1194 2     BEGIN
619 1195 2     SIGNAL(msg(netasn), 0, .status);    ! then signal the error

```

```

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

```

```

.PSECT $PLITS,NOWRT,NOEXE,2
00 00 04 03 00055 P.AAL: .BYTE 3, 4, 0, 0
00 00 00 3A 54 45 4E 5F 00059 .BLKB 3
010E0005, 00064 P.AAN: .ASCII \ NET:\<0><0><0>
00000000, 00068 P.AAM: .LONG 17694725
.ADDRESS P.AAN

```

```

.EXTRN SYSS$ASSIGN, EVL$_LOGOPNR
.PSECT $CODE$,NOWRT,2

```

```

003C 00000 OPEN_EVENT LINK:
55 00000000G 00 9E 00002 .WORD Save R2,R3,R4,R5 : 1139
54 00000000G 8F D0 00009 MOVAB LIB$SIGNAL, R5
5E 04 C2 00010 MOVL #EVL$_NETASN, R4
52 04 AC D0 00013 SUBL2 #4, SP
MOVLE 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), RO	1175
15	A2		50		2F	3A	00024	LOCC	#47, RO, 21(R2)	
					02	12	00029	BNEQ	1\$	
					51	D4	0002B	CLRL	R1	
					51	D5	0002D	TSTL	PTR	1177
					09	13	0002F	BEQL	2\$	
14	A2		50	15	A2	9E	00031	MOVAB	21(R2), RO	1179
			51		50	83	00035	SUBB3	RO, PTR, 20(R2)	
		03	A1		CF	D0	0003A	MOVL	P.AAL, 3(PTR)	1187
					7E	7C	00040	CLRQ	-(SP)	1190
					0A	A2	9F	00042	PUSHAB	10(R2)
					0A	CF	9F	00045	PUSHAB	P.AAM
		00000000G	00		04	FB	00049	CALLS	#4, SYSS\$ASSIGN	
			53		50	D0	00050	MOVL	RO, 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)
			7E		32	DD	00064	PUSHL	#50	
					A2	3C	00066	MOVZWL	10(R2), -(SP)	
00000000G			00		01	DD	0006A	PUSHL	#1	
			53		0C	FB	0006C	CALLS	#12, SYSS\$QIOW	
			07		50	D0	00073	MOVL	RO, STATUS	
			53		53	E9	00076	BLBC	STATUS, 3\$	1206
			11		0C	A2	3C	00079	MOVZWL	12(R2), STATUS
					53	E8	0007D	BLBS	STATUS, 4\$	1208
					53	DD	00080	PUSHL	STATUS	1210
					7E	D4	00082	CLRL	-(SP)	1213
					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	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	
			65	00000000G	8F	DD	000A5	PUSHL	#EVL\$ LOGOPNR	
					05	FB	000AB	CALLS	#5, LIB\$SIGNAL	
					52	DD	000AE	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 2     BEGIN
684 1259 2     status = $DASSGN(CHAN = .iec [iec$w_chan]); ! Deassign network channel
685 1260 2
686 1261 2     IF NOT .status          ! If error detected,
687 1262 2     THEN
688 1263 2     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 SYSSDASSGN, EVL\$_NETDAS

001C 0000 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, SYSSDASSGN	:
	53		50	D0 0001E	MOVL	R0, STATUS	:

RECEIVER
V04-000

M 15
16-Sep-1984 01:37:57
14-Sep-1984 12:28:54

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EVL.SRC]RECEIVER.B32;1

Page 25
(9)

	11		53	E8	00021	BLBS	STATUS, 1\$:	1261
			53	DD	00024	PUSHL	STATUS	:	1263
			7E	D4	00026	CLRL	-(SP)	:	
00000000G	00	00000000G	8F	DD	00028	PUSHL	#EVL\$ NETDAS	:	
	53		03	FB	0002E	CALLS	#3, LIB\$SIGNAL	:	
	50		62	OF	00035	REMQUE	(R2), STATUS	:	1266
	6E	04	AC	D0	00038	MOVL	IEC, R0	:	1267
		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	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	:	
0000' CF	53		52	D2	00067	MCOML	R2, R3	:	
	50		53	8B	0006A	BICB3	R3, R0, EVL\$B_RCVDONE	:	
			04	00070	RET			:	1273

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

RE
VO

```

: 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 2 BEGIN
: 727 1301 3 next_ptr = .ptr [iec$I_link];
: 728 1302 3 close_event_link(.ptr); ! Abort the incoming link
: 729 1303 3 ptr = .next_ptr; ! and link to next in chain
: 730 1304 2 END;
: 731 1305 2
: 732 1306 2 ptr = .sink_header; ! Start at first sink block
: 733 1307 2 WHILE .ptr NEQ sink_header ! Until end of linked list,
: 734 1308 2 DO
: 735 1309 2 BEGIN
: 736 1310 3 ptr [sink$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$add_work_item(delete_sink,.ptr); ! and delete the sink
: 739 1313 3 ptr = .ptr [sink$I_link]; ! Link to next one
: 740 1314 2 END;
: 741 1315 2
: 742 1316 1 END;

```

001C 0000 SHUTDOWN:

54	0000'	CF	9E	00002	.WORD	Save R2,R3,R4	: 1274
52		64	00	00007	MOVAB	IEC_HEADER, R4	: 1297
50		64	9E	0000A	MOVL	IEC_HEADER, PTR	: 1298
50		52	D1	0000D	MOVAB	IEC_HEADER, R0	: 1301
		0F	13	00010	CPL	PTR, R0	: 1302
53		62	00	00012	BEQL	2\$	
		52	DD	00015	MOVL	(PTR), NEXT_PTR	
					PUSHL	PTR	

B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~

RECEIVER
V04-000

B 16
16-Sep-1984 01:37:57
14-Sep-1984 12:28:54

VAX-11 Bliss-32 V4.0-742
DISK\$VM\$MASTER:[EVL.SRC]RECEIVER.B32;1 (10) Page 27

FF73	CF		01	FB	00017	CALLS	#1, CLCSE_EVENT_LINK	:		
	52		53	DO	0001C	MOVL	NEXT_PTR, PTR	:	1303	
			E9	11	0001F	BRB	1\$:	1298	
	52	FB	A4	DO	00021	2\$:	MOVL	SINK_HEADER, PTR	:	1306
	50	FB	A4	9E	00025	3\$:	MOVAB	SINK_HEADER, RO	:	1307
	50		52	D1	00029		CMPL	PTR, RO	:	
			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	DO	00048		MOVL	(PTR), PTR	:	1313
			D8	11	0004B		BRB	3\$:	1307
			04	0004D	4\$:	RET		:	1316	

; Routine Size: 78 bytes. Routine Base: \$CODE\$ + 04C0

```

: 744 1317 1 ROUTINE wait_for_event (iec): NOVALUE =
: 745 1318 1
: 746 1319 1 ---
: 747 1320 1
: 748 1321 1 This routine is called to obtain processed events from
: 749 1322 1 the receiver's incoming logical link. Each incoming event
: 750 1323 1 is immediately queued to the disposal queue for the appropriate
: 751 1324 1 sink types.
: 752 1325 1
: 753 1326 1 Inputs:
: 754 1327 1
: 755 1328 1 iec = Address of incoming event channel context block
: 756 1329 1
: 757 1330 1 Outputs:
: 758 1331 1
: 759 1332 1 None
: 760 1333 1 ---
: 761 1334 1
: 762 1335 2 BEGIN
: 763 1336 2
: 764 1337 2 MAP
: 765 1338 2 iec: REF BBLOCK; ! Address of incoming event channel
: 766 1339 2
: 767 1340 2 LOCAL
: 768 1341 2 status;
: 769 1342 2
: 770 P 1343 2 status = $QIO(FUNC = IOS_READVBLK, ! Get event from incoming link
: 771 P 1344 2 CHAN = .iec [iec$w_chan],
: 772 P 1345 2 EFN = evl$e_async_efn,
: 773 P 1346 2 IOSB = iec [iec$w_iosb], ! Address of I/O status block
: 774 P 1347 2 ASTADR = event_received, ! Address of completion routine
: 775 P 1348 2 ASTPRM = .iec, ! Giving IEC as routine parameter
: 776 P 1349 2 P1 = iec [iec$t_event], ! Address of event buffer
: 777 1350 2 P2 = iec$e_maxevtlen); ! Length of event buffer
: 778 1351 2
: 779 1352 2 IF NOT .status ! If unsuccessful
: 780 1353 2 THEN
: 781 1354 3 BEGIN
: 782 1355 3 SIGNAL(msg(readevt),0,.status); ! then signal error
: 783 1356 3 close_event_link(.iec); ! close link; deallocate storage
: 784 1357 2 END;
: 785 1358 2
: 786 1359 1 END;

```

```

0004 0000 WAIT_FOR_EVENT:
: 7E 7C 00002 :WORD Save R2 : 1317
: 7E 7C 00004 CLRQ -(SP) : 1350
: 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

```

RECEIVER
V04-000

		OC	A2	9F	00017	PUSHAB	12(R2)	:
			31	DD	0001A	PUSHL	#49	:
	7E	OA	A2	3C	0001C	MOVZWL	10(R2), -(SP)	:
			02	DD	00020	PUSHL	#2	:
00000000G	00		0C	FB	00022	CALLS	#12, SYSSQIO	:
	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

```

: 788      1360 1 ROUTINE event_received (iec): NOVALUE =
: 789      1361 1
: 790      1362 1
: 791      1363 1
: 792      1364 1
: 793      1365 1
: 794      1366 1
: 795      1367 1
: 796      1368 1
: 797      1369 1
: 798      1370 1
: 799      1371 1
: 800      1372 1
: 801      1373 1
: 802      1374 1
: 803      1375 1
: 804      1376 1
: 805      1377 2
: 806      1378 2
: 807      1379 2
: 808      1380 2
: 809      1381 2
: 810      1382 2
: 811      1383 2
: 812      1384 3
: 813      1385 3
: 814      1386 3
: 815      1387 3
: 816      1388 3
: 817      1389 3
: 818      1390 2
: 819      1391 2
: 820      1392 2
: 821      1393 2
: 822      1394 2
: 823      1395 2
: 824      1396 2
: 825      1397 2
: 826      1398 1

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

```

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

RECEIVER
V04-000

F 16
16-Sep-1984 01:37:57
14-Sep-1984 12:28:54

VAX-11 Bliss-32 V4.0-742
DISK\$VMMASTER:[EVL.SRC]RECEIVER.B32;1 (12) Page 31

		55	A2	9F	0002D	28:	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

```

828 1399 1 GLOBAL ROUTINE evl$queue_event (length, event): NOVALUE =
829 1400 1
830 1401 1 ---
831 1402 1
832 1403 1 This routine is called to queue a given event record
833 1404 1 to the appropriate sink type(s).
834 1405 1
835 1406 1 Inputs:
836 1407 1
837 1408 1 length = Length of event record
838 1409 1 event = Address of event record:
839 1410 1
840 1411 1 0) Function code (always 1 = event record)
841 1412 1 1) Event flags (byte) which sink(s) get the event
842 1413 1 2) Event code (word)
843 1414 1 4) Time of event (6 bytes) in Julian half-day format
844 1415 1 10) Source node (2 + 1-7 bytes) address and name
845 1416 1 ?) Event entity (1+n bytes) describing line or node ID
846 1417 1 ?) Event data, depending on event code
847 1418 1
848 1419 1 Outputs:
849 1420 1
850 1421 1 None
851 1422 1 ---
852 1423 1
853 1424 2 BEGIN
854 1425 2
855 1426 2 MAP
856 1427 2 event: REF BBLOCK; ! Address of event record
857 1428 2
858 1429 2 LOCAL
859 1430 2 event_overflow: BYTE, ! True if event queue overflowed
860 1431 2 sink: REF BBLOCK; ! Address of sink descriptor block
861 1432 2
862 1433 2 IF .evl$gl_logmask [elg$v_rcvevt] ! If logging received messages,
863 1434 2 THEN
864 1435 3 BEGIN
865 1436 3 LOCAL msgdesc: VECTOR [2];
866 1437 3 msgdesc [0] = .length; ! Setup descriptor of message
867 1438 3 msgdesc [1] = .event;
868 1439 3 EVL$PRINTLOG($BITPOSITION(elg$v_rcvevt), ! Log the received message
869 1440 3 %ASCID 'Event received',0,msgdesc);
870 1441 3 END;
871 1442 2
872 1443 2 event_overflow = false; ! Preset no overflow
873 1444 2 sink = .sink_header [0]; ! Start at first sink entry
874 1445 2
875 1446 2 WHILE .sink NEQ sink_header ! Until end of sink list
876 1447 2 DO
877 1448 3 BEGIN
878 1449 3 IF .(event [evt$b_flags])<.sink [sink$b_type]-1,1> ! If this sink gets event,
879 1450 3 AND .sink [sink$b_state] NEQ sink$c_off ! and delete not pending,
880 1451 3 THEN
881 1452 4 BEGIN ! then insert into sink's queue
882 1453 4 LOCAL
883 1454 4 status,
884 1455 4 entry: REF BBLOCK; ! Address of event data block

```


RECEIVER
V04-000

I 16
16-Sep-1984 01:37:57
14-Sep-1984 12:28:54

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EVL.SRC]RECEIVER.B32;1
Page 34 (13)

		0A	50		6E	D0	0005B	MOVL	ENTRY, R0	:	1459
		08	A0	04	AC	B0	0005E	MOVW	LENGTH, 10(R0)	:	
0C	A0		BC	04	AC	28	00063	MOV3	LENGTH, @EVENT, 12(R0)	:	1460
			0B		58	E9	0006A	BLBC	STATUS, 3\$:	1461
					56	DD	0006D	PUSHL	SINK	:	1463
		0000G	CF	0000V	CF	9F	0006F	PUSHAB	OUTPUT_EVENTS	:	
			0A	0A	02	FB	00073	CALLS	#2, WKQ\$ADD_WORK_ITEM	:	
					A6	B1	00078	CMPW	10(SINK), #T0	:	1464
					03	1F	0007C	BLSSU	4\$:	
			59		01	90	0007E	MOVB	#1, EVENT_OVERFLOW	:	1466
			56		66	D0	00081	MOVL	(SINK), SINK	:	1468
					A5	11	00084	BRB	2\$:	1446
						04	00086	RET		:	1471

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

```

1472 1 ROUTINE output_events (sink): NOVALUE =
1473 1
1474 1 ---
1475 1
1476 1 This routine is called when an event is queued for
1477 1 any sink. All events for the given sink are written
1478 1 to the appropriate destination depending on the sink
1479 1 type. Note that this routine may be called when
1480 1 there are no events queued if the sink is turned off
1481 1 and wants to be cleaned up here.
1482 1
1483 1 Inputs:
1484 1
1485 1 sink = Address of sink descriptor block
1486 1
1487 1 Outputs:
1488 1
1489 1 None
1490 1 ---
1491 1
1492 2 BEGIN
1493 2
1494 2 MAP
1495 2 sink: REF BBLOCK; ! Address of sink block
1496 2
1497 2 LOCAL
1498 2 event_data: REF BBLOCK; ! Address of next event data block
1499 2 event: REF BBLOCK; ! Address of actual event record
1500 2
1501 2 IF .sink [sink$b_state] EQL sink$c_hold ! If holding sink output,
1502 2 THEN
1503 2 RETURN; ! return without dequeuing anything
1504 2
1505 2 WHILE NOT REMQUE(.sink [sink$l_evtfl], event_data) ! Dequeue the next event
1506 2 DO
1507 2 BEGIN
1508 2 sink [sink$w_events] = .sink [sink$w_events] - 1; ! Decrement events left
1509 2 event = event_data [evq$t_event]; ! Address the actual event record
1510 2
1511 2 IF .sink [sink$b_state] EQL sink$c_on ! If sink is ON,
1512 2 THEN
1513 2 CASE .sink [sink$b_type] FROM 0 TO sink$c_monitor ! Dispatch to routine
1514 2 OF
1515 2 SET
1516 2 [sink$c_console]: output_console( ! Output to console
1517 2 .sink, .event, .event_data [evq$w_evtsize]);
1518 2 [sink$c_file]: output_file( ! Output to file
1519 2 .sink, .event, .event_data [evq$w_evtsize]);
1520 2 [sink$c_monitor]:
1521 2 BEGIN
1522 2 output_opcom( ! Output to OPCOM terminals
1523 2 .sink, .event, .event_data [evq$w_evtsize]);
1524 2 output_monitor( ! Output to process (if any)
1525 2 .sink, .event, .event_data [evq$w_evtsize]);
1526 2 END;
1527 2 [INRANGE]: SIGNAL(msg(badtype), 1, .sink [sink$b_type]);
1528 2 TES;

```

```

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

```

```

                                .EXTRN  EVL$_BADTYPE
                                001C 00000 OUTPUT_EVENTS:
                                .WORD  Save R2,R3,R4
                                50      04  AC  D0 00002      MOVL  SINK, R0      : 1472
                                02      09  A0  91 00006      CMPB  9(R0), #2      : 1501
                                01      12  0000A      BNEQ  1$
                                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)
                                5A      12  00025      BNEQ  8$
                                00      08  A4  RF 00027      CASEB 8(R4), #0, #3      : 1513
0026      03      00      00      0042      0002C 3$:      .WORD  7$-3$,-
                                4$-3$,-
                                5$-3$,-
                                6$-3$
                                7E      0A  A2  3C 00034 4$:      MOVZWL 10(EVENT_DATA), -(SP)      : 1517
                                53      DD  00038      PUSHL  EVENT
                                54      DD  0003A      PUSHL  R4
                                0000V CF      03  FB 0003C      CALLS  #3, OUTPUT_CONSOLE
                                7E      0A  A2  3C 00043 5$:      MOVZWL 10(EVENT_DATA), -(SP)      : 1516
                                53      DD  00047      PUSHL  EVENT
                                54      DD  00049      PUSHL  R4
                                0000V CF      03  FB 0004B      CALLS  #3, OUTPUT_FILE
                                7E      0A  A2  3C 00052 6$:      MOVZWL 10(EVENT_DATA), -(SP)      : 1518
                                53      DD  00056      PUSHL  EVENT
                                54      DD  00058      PUSHL  R4
                                0000V CF      03  FB 0005A      CALLS  #3, OUTPUT OPCOM
                                7E      0A  A2  3C 0005F      MOVZWL 10(EVENT_DATA), -(SP)      : 1525
                                53      DD  00063      PUSHL  EVENT
                                54      DD  00065      PUSHL  R4
                                0000V CF      03  FB 00067      CALLS  #3, OUTPUT_MONITOR
                                7E      08  A4  9A 0006E 7$:      MOVZBL 8(R4), -(SP)      : 1513
                                01      DD  00072      PUSHL  #1      : 1527
                                00000000G 8F  DD  00074      PUSHL  #EVL$_BADTYPE

```

RECEIVER
V04-000

L 16
16-Sep-1984 01:37:57
14-Sep-1984 12:28:54

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EVL.SRC]RECEIVER.B32;1 Page 37 (14)

00000000G	00	03	FB	0007A	CALLS	#3, LIB\$SIGNAL	:	
		52	DD	00081	PUSHL	EVENT DATA	:	1530
0000G	CF	01	FB	00083	CALLS	#1, EVL\$DEALLOCDBK	:	
		87	11	00088	BRB	2\$:	1505
	50	04	AC	D0 0008A	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

```

: 971 1540 1 ROUTINE output_console (sink, event, event_length): NOVALUE =
: 972 1541 1
: 973 1542 1 ---
: 974 1543 1
: 975 1544 1 Output an event record to the console sink.
: 976 1545 1
: 977 1546 1 Inputs:
: 978 1547 1
: 979 1548 1 sink = Address of sink descriptor block
: 980 1549 1 event = Address of event record
: 981 1550 1 event_length = Length of event record
: 982 1551 1
: 983 1552 1 Outputs:
: 984 1553 1
: 985 1554 1 None
: 986 1555 1 ---
: 987 1556 1
: 988 1557 2 BEGIN
: 989 1558 2
: 990 1559 2 MAP
: 991 1560 2 sink: REF BBLOCK, ! Address of sink block
: 992 1561 2 event: REF BBLOCK; ! Address of event record
: 993 1562 2
: 994 1563 2 LOCAL
: 995 1564 2 status,
: 996 1565 2 desc: VECTOR [2]; ! Descriptor
: 997 1566 2
: 998 1567 2 IF .sink [sink$l_closertn] EQL 0 ! If sink not yet initialized,
: 999 1568 2 THEN
: 1000 1569 2 BEGIN
: 1001 1570 2 IF .sink [sink$b_namelen] EQL 0 ! If no sink name specified,
: 1002 1571 2 THEN
: 1003 1572 2 BEGIN
: 1004 1573 2 sink [sink$b_namelen] = 6; ! then default to _OPA0:
: 1005 1574 2 CHSMOVE(6, UPLIT BYTE('_OPA0:'), sink [sink$t_name]);
: 1006 1575 2 END;
: 1007 1576 2 open_file(.sink); ! then open the file/device
: 1008 1577 2 IF .sink [sink$l_closertn] EQL 0 ! If unsuccessful,
: 1009 1578 2 THEN
: 1010 1579 2 RETURN; ! then forget about the output
: 1011 1580 2 END;
: 1012 1581 2
: 1013 1582 2 format_event(.event, .event_length, write_line, .sink); ! Format the event
: 1014 1583 2
: 1015 1584 2 desc [0] = 0;
: 1016 1585 2 write_line(.sink, desc); ! write a null line afterwards
: 1017 1586 2
: 1018 1587 1 END;

```

.PSECT SPLITS,NOWRT,NOEXE,2

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

:

.PSECT \$CODE\$,NOWRT,2

		007C 0000 OUTPUT_CONSOLE:					
	SE	08	C2	00002	.WORD	Save R2,R3,R4,R5,R6	1540
	56	04	AC	D0	SUBL2	#8, SP	1567
		22	A6	D5	MOVL	SINK, R6	
			1C	12	TSTL	34(R6)	
		2E	A6	95	BNEQ	2\$	
			0B	12	TSTB	46(R6)	1570
			06	90	BNEQ	1\$	
2F	A6	2E	A6	06	MOVB	#6, 46(R6)	1573
		0000	CF	06	MOVCL	#6, P.AAQ, 47(R6)	1574
		0000V	CF	56	PJSHL	R6	1576
				01	CALLS	#1, OPEN_FILE	
				22	TSTL	34(R6)	1577
				1A	BEQL	3\$	
				56	PUSHL	R6	1582
		0000V	CF	9F	PUSHAB	WRITE_LINE	
			7E	08	MOVQ	EVENT, -(SP)	
		0000G	CF	04	CALLS	#4, FORMAT_EVENT	
				6E	CLRL	DESC	1584
				4040	PUSHR	#*M<R6, SP>	1585
		0000V	CF	02	CALLS	#2, WRITE_LINE	
				04	RET		1587

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

```

1020 1588 1 ROUTINE write_line (sink, bufdesc): NOVALUE =
1021 1589 1
1022 1590 1 ---
1023 1591 1
1024 1592 1 This routine is called at the end of every line
1025 1593 1 in the formatted display. The line is written
1026 1594 1 to the device or file specified by the sink.
1027 1595 1 If we are logging received events to the batch
1028 1596 1 job log, write the formatted text to the log as well.
1029 1597 1
1030 1598 1 Inputs:
1031 1599 1
1032 1600 1 sink = Address of sink control block
1033 1601 1 bufdesc = Address of descriptor of current output buffer
1034 1602 1
1035 1603 1 Outputs:
1036 1604 1
1037 1605 1 The line is written.
1038 1606 1 ---
1039 1607 1
1040 1608 2 BEGIN
1041 1609 2
1042 1610 2 MAP
1043 1611 2 sink: REF BBLOCK; ! Address of sink control block
1044 1612 2
1045 1613 2 BIND
1046 1614 2 desc = .bufdesc: VECTOR, ! Address line descriptor
1047 1615 2 timeout = UPLIT(-15*10*1000*1000,-1); ! I/O timeout = 15 seconds
1048 1616 2
1049 1617 2 LOCAL
1050 1618 2 status,
1051 1619 2 fab: REF BBLOCK, ! Address of FAB
1052 1620 2 rab: REF BBLOCK; ! Address of RAB
1053 1621 2
1054 1622 2 IF .evl$gl_logmask [elg$v_rcvevt] ! If logging received events,
1055 1623 2 THEN
1056 1624 2 LIB$PUT_OUTPUT(desc); ! then write line to batch job log
1057 1625 2
1058 1626 2 IF .sink [sink$l_closetrn] EQL 0 ! If file no longer open,
1059 1627 2 THEN
1060 1628 2 RETURN; ! then skip it
1061 1629 2
1062 1630 2 rab = .sink [sink$l_rab]; ! Get address of RAB
1063 1631 2 fab = .rab [rab$l_fab]; ! Get address of FAB
1064 1632 2
1065 1633 2 rab [rab$w_rsz] = .desc [0]; ! Set up length/address of this line
1066 1634 2 rab [rab$l_rbf] = .desc [1];
1067 1635 2
1068 1636 2 status = $PUT(RAB = .rab); ! Output the record
1069 1637 2
1070 1638 2 .F NOT .status ! If error detected,
1071 1639 2 THEN
1072 1640 2 BEGIN
1073 1641 2 LOCAL name: VECTOR [2];
1074 1642 2 name [0] = .sink [sink$b_namelen];
1075 1643 2 name [1] = sink [sink$t_name];
1076 1644 2 SIGNAL(msg(writefile), T, name, .status, .rab [rab$l_stv]);

```

RECEIVER
V04-000

: 1077 1645 3 sink [sink\$V_error] = true;
: 1078 1646 3
: 1079 1647 2 END;
: 1080 1648 2
: 1081 1649 1 END;

. Suspend all operations on sink
. until data base change

.PSECT \$SPLITS,NOWRT,NOEXE,2
000BA .BLKB 2
FFFFFFF F70F2E80 000BC P.AAR: .LONG -150000000, -1
TIMEOUT= P.AAR
.EXTRN SYSS\$PUT, EVL\$WRITEFILE
.PSECT \$CODE\$,NOWRT,2

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

: Routine Size: 99 bytes. Routine Base: \$CODE\$ + 0708

```

: 1083 1650 1 ROUTINE output_file (sink, event, event_length): NOVALUE =
: 1084 1651 1
: 1085 1652 1 |---
: 1086 1653 1 |
: 1087 1654 1 |         Output an event record to a file.
: 1088 1655 1 |
: 1089 1656 1 |     Inputs:
: 1090 1657 1 |
: 1091 1658 1 |         sink = Address of sink descriptor block
: 1092 1659 1 |         event = Address of event record
: 1093 1660 1 |         event_length = Length of event record
: 1094 1661 1 |
: 1095 1662 1 |     Outputs:
: 1096 1663 1 |
: 1097 1664 1 |         None
: 1098 1665 1 |---
: 1099 1666 1
: 1100 1667 2 BEGIN
: 1101 1668 2
: 1102 1669 2 MAP
: 1103 1670 2     sink:      REF BBLOCK,      ! Address of sink block
: 1104 1671 2     event:    REF BBLOCK;    ! Address of event record
: 1105 1672 2
: 1106 1673 2 LOCAL
: 1107 1674 2     status,
: 1108 1675 2     name:      VECTOR [2],    ! Descriptor of file name
: 1109 1676 2     rab:      REF BBLOCK;    ! Address of RAB
: 1110 1677 2
: 1111 1678 2 IF .sink [sink$v_error]      ! If in error state,
: 1112 1679 2 THEN
: 1113 1680 2     RETURN;                    ! then don't do anything
: 1114 1681 2
: 1115 1682 2 name [0] = .sink [sink$b_namelen]; ! Setup descriptor of file name
: 1116 1683 2 name [1] = sink [sink$t_name];
: 1117 1684 2
: 1118 1685 2 IF .sink [sink$l_closetrn] EQL 0 ! If sink is not yet initialized
: 1119 1686 2 THEN
: 1120 1687 2     open_file(.sink);          ! then open the file
: 1121 1688 2
: 1122 1689 2 rab = .sink [sink$l_rab];        ! Get address of RAB
: 1123 1690 2 rab [rab$w_rsz] = .event_length; ! Set length of event record
: 1124 1691 2 rab [rab$l_rbf] = .event;      ! and address
: 1125 1692 2
: 1126 1693 2 status = $PUT(RAB = .rab);      ! Write event record to file
: 1127 1694 2
: 1128 1695 2 IF NOT .status                  ! If error detected,
: 1129 1696 2 THEN
: 1130 1697 2     BEGIN
: 1131 1698 2     SIGNAL(msg(writefile), 1, name, .status, .rab [rab$l_stv]);
: 1132 1699 2     close_file(.sink);          ! (close sink file
: 1133 1700 2     sink [sink$v_error] = true; ! Suspend all operations on sink
: 1134 1701 2     ! until data base change
: 1135 1702 2     END;
: 1136 1703 2
: 1137 1704 1 END;

```

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

; Routine Size: 96 bytes. Routine Base: \$CODE\$ + 076B

```
1139 1705 1 GLOBAL ROUTINE open_file (sink): NOVALUE =
1140 1706 1
1141 1707 1 |---
1142 1708 1 |
1143 1709 1 |       Open a file sink.
1144 1710 1 |
1145 1711 1 | Inputs:
1146 1712 1 |
1147 1713 1 |       sink = Address of sink control block
1148 1714 1 |
1149 1715 1 | Outputs:
1150 1716 1 |
1151 1717 1 |       None
1152 1718 1 |---
1153 1719 1
1154 1720 2 BEGIN
1155 1721 2
1156 1722 2 MAP
1157 1723 2     sink:          REF BBLOCK;          ! Address of sink block
1158 1724 2
1159 1725 2 LOCAL
1160 1726 2     status,
1161 1727 2     length,          ! Length of FAB and RAB storage
1162 1728 2     name:            ! Descriptor of sink name
1163 1729 2     rab:            REF BBLOCK;          ! Address of RAB
1164 1730 2     fab:            REF BBLOCK;          ! Address of FAB
1165 1731 2
1166 1732 2     name [0] = .sink [sink$b_namelen]; ! Setup descriptor of file name
1167 1733 2     name [1] = sink [sink$t_name];
1168 1734 2
1169 1735 2     length = rab$c_bln + fab$c_bln;
1170 1736 2     signal_if_error(LIB$GET_VM(length,rab)); ! Allocate RAB/FAB
1171 1737 2
1172 1738 2     fab = .rab + rab$c_bln;          ! Get address of FAB
1173 1739 2
1174 1740 2 $FAB_INIT(FAB = .fab,          ! Initialize FAB
1175 1741 2     FNS = .name [0],
1176 1742 2     FNA = .name [1],
1177 1743 2     DNM = '.LOG',
1178 1744 2     FAC = PUT,
1179 1745 2     RAT = CR,
1180 1746 2     FOP = (CIF,SQO),
1181 1747 2     SHR = (GET,UPI));
1182 1748 2 $RAB_INIT(RAB = .rab,          ! Initialize RAB
1183 1749 2     FAB = .fab,
1184 1750 2     ROP = (EOF,WBH));
1185 1751 2
1186 1752 2     status = $CREATE(FAB = .fab); ! Create the event file
1187 1753 2
1188 1754 2     IF NOT .status          ! If error detected,
1189 1755 2     THEN
1190 1756 2     BEGIN
1191 1757 2     IF .fab [fab$l_stv] NEQ ss$_devalloc ! If not "device allocated" error,
1192 1758 2     THEN                                ! (which is ok, since user might want
1193 1759 2     ! to use the output terminal)
1194 1760 2     SIGNAL(msg(openfile), 1, name, .status, .fab [fab$l_stv]);
1195 1761 2     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_close rtn] = close_file; ! Set file opened
: 1210      1776 2
: 1211      1777 1   END;

```

.PSECT \$SPLITS,NOWRT,NOEXE,2

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

.EXTRN SYSS\$CREATE, EVL\$_OPENFILE
.EXTRN SYSS\$CONNECT

.PSECT \$CODE\$,NOWRT,2

.ENTRY OPEN FILE, Save R2,R3,R4,R5,R6,R7,R8,R9,R10 ; 1705

	5A	G0000000G	8F	D0	00002	MOV	R0, #EVL\$_OPENFILE, R10	
	59	00000000G	00	9E	00009	MOVAB	LIB\$SIGNAL, R9	
	5E		10	C2	00010	SUBL2	#16, SP	
	58	04	AC	D0	00013	MOV	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	MOV	R0, STATUS	
	06		52	E8	00035	BLBS	STATUS, 1\$	
			52	DD	00038	PUSHL	STATUS	
	69		01	FB	0003A	CALLS	#1, LIB\$SIGNAL	
			04	00	0003D	RET		
	57		6	D0	0003E	MOV	RAB, R7	1738
	56	44	A7	9E	00041	MOVAB	68(R7), FAB	
0050	8F	00	00	2C	00045	MOVCS	#0, (SP), #0, #80, (FAB)	1747
			66		0004C			
	66	5003	8F	B0	0004D	MOVW	#20483, (FAB)	
04	A6	02000040	8F	D0	00052	MOV	#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	MOV	NAME+4, 44(FAB)	
30	A6	0000	CF	9E	0006B	MOVAB	P.AAS, 48(FAB)	
34	A6	08	AE	90	00071	MOV	NAME, 52(FAB)	
0044	8F	00	04	90	00076	MOV	#4, 53(FAB)	
			00	2C	0007A	MOVCS	#0, (SP), #0, #68, (R7)	1750

		67	00081						
		8F B0	00082	MOVW	#17409, (R7)				
	04 A7	0500 8F 3C	00087	MOVZWL	#1280, 4(R7)				
	3C A7		56 D0	MOVL	FAB, 60(R7)				
			56 DD	PUSHL	FAB			1752	
00000000G	00		01 FB	CALLS	#1, SYSS\$CREATE				
	53		50 D0	MOVL	R0, STATUS				
	0F		53 E8	BLBS	STATUS, 2\$			1754	
00000840	8F	0C	A6 D1	CMP	12(FAB), #2112			1757	
			26 13	BEQL	4\$				
		0C	A6 DD	PUSHL	12(FAB)			1760	
			15 11	BRB	3\$				
	52		6E D0	MOVL	RAB, R2			1765	
			52 DD	PUSHL	R2				
00000000G	00		01 FB	CALLS	#1, SYSS\$CONNECT				
	53		50 D0	MOVL	R0, STATUS				
	1C		53 E8	BLBS	STATUS, 5\$			1766	
		0C	A2 DD	PUSHL	12(R2)			1769	
			53 DD	PUSHL	STATUS				
		10	AE 9F	PUSHAB	NAME				
			01 DD	PUSHL	#1				
			5A DD	PUSHL	R10				
	69		05 FB	CALLS	#5, LIB\$SIGNAL				
			5E DD	PUSHL	SP			1770	
		08	AE 9F	PUSHAB	LENGTH				
00000000G	00		02 FB	CALLS	#2, LIB\$FREE_VM				
			04	RET				1768	
	1E A8		6E D0	MOVL	RAB, 30(R8)			1774	
	22 A8	0000V	CF 9E	MOVAB	CLOSE_FILE, 34(R8)			1775	
			04	RET				1777	

; Routine Size: 232 bytes. Routine Base: \$CODE\$ + 07CB


```

1213 1778 1 ROUTINE close_file (sink): NOVALUE =
1214 1779 1
1215 1780 1 |---
1216 1781 1 |
1217 1782 1 |         Close a file sink.
1218 1783 1 |
1219 1784 1 | Inputs:
1220 1785 1 |
1221 1786 1 |         sink = Address of sink block
1222 1787 1 |
1223 1788 1 | Outputs:
1224 1789 1 |
1225 1790 1 |         None
1226 1791 1 |---
1227 1792 1
1228 1793 2 BEGIN
1229 1794 2
1230 1795 2 MAP
1231 1796 2     sink:          REF BBLOCK;          ! Address of sink block
1232 1797 2
1233 1798 2 LOCAL
1234 1799 2     status,
1235 1800 2     rab:          REF BBLOCK;          ! Address of RAB
1236 1801 2
1237 1802 2     rab = .sink [sink$l_rab];          ! Get address of RAB
1238 1803 2
1239 1804 2     IF .rab NEQ 0          ! If RAB and FAB were allocated
1240 1805 2     THEN
1241 1806 2         BEGIN
1242 1807 2             LOCAL
1243 1808 2                 length,
1244 1809 2                 name:  VECTOR [2],          ! Descriptor of file name
1245 1810 2                 fab:   REF BBLOCK;          ! Address of FAB
1246 1811 2
1247 1812 2                 fab = .rab [rab$l_fab];          ! Get address of FAB
1248 1813 2
1249 1814 2                 name [0] = .sink [sink$b_namelen]; ! Setup descriptor of file name
1250 1815 2                 name [1] = sink [sink$t_name];
1251 1816 2
1252 1817 2                 $DISCONNECT(RAB = .rab);          ! Disconnect to cancel any I/O
1253 1818 2
1254 1819 2                 status = $CLOSE(FAB = .fab);      ! Close the file
1255 1820 2
1256 1821 2                 IF NOT .status          ! If error detected,
1257 1822 2                 THEN
1258 1823 2                     SIGNAL(msg(closefile), 1, name, .status, .fab [fab$l_stv]);
1259 1824 2
1260 1825 2                 length = rab$c_bln + fab$c_bln;    ! Length of RAB/FAB storage
1261 1826 2                 LIB$FREE_VM(length,rab);          ! Deallocate RAB and FAB
1262 1827 2                 END;
1263 1828 2
1264 1829 2     sink [sink$l_closertn] = 0;          ! Set sink no longer open
1265 1830 2
1266 1831 1 END;

```

.EXTRN SYSSDISCONNECT, SYSSCLOSE
.EXTRN EVL\$ _CLOSEFILE

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

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

```
1268 1832 1 ROUTINE output_opcom (sink, event, event_length): NOVALUE =
1269 1833 1
1270 1834 1 ---
1271 1835 1
1272 1836 1 Output a formatted event to all OPCOM network operators
1273 1837 1
1274 1838 1 Inputs:
1275 1839 1
1276 1840 1 sink = Address of sink descriptor block
1277 1841 1 event = Address of event record
1278 1842 1 event_length = Length of event record
1279 1843 1
1280 1844 1 Outputs:
1281 1845 1
1282 1846 1 None
1283 1847 1 ---
1284 1848 1
1285 1849 2 BEGIN
1286 1850 2
1287 1851 2 MAP
1288 1852 2 sink: REF BBLOCK, ! Address of sink block
1289 1853 2 event: REF BBLOCK; ! Address of event record
1290 1854 2
1291 1855 2 LOCAL
1292 1856 2 status,
1293 1857 2 opcom_desc: VECTOR [2], ! Descriptor of OPCOM buffer
1294 1858 2 opcom_buf: BBLOCK [$BYTEOFFSET(opc$l_ms_text)+2048];
1295 1859 2
1296 1860 2 !
1297 1861 2 ! Always write the formatted event to all OPCOM network operators,
1298 1862 2 ! whether a monitor task is specified or not. This allows monitor
1299 1863 2 ! tasks to receive and process events, and we still get the OPCOM feature.
1300 1864 2 !
1301 1865 2
1302 1866 2 sink [sink$w_maxbufsiz] = 2048; ! Set maximum limit for output
1303 1867 2 sink [sink$w_buflen] = 0; ! Initialize buffer descriptor
1304 1868 2 sink [sink$l_buffer] = opcom_buf + $BYTEOFFSET(opc$l_ms_text);
1305 1869 2
1306 1870 2 format_event(.event, .event_length, ! Call formatting routines
1307 1871 2 save_lines, .sink); ! but append output lines together
1308 1872 2
1309 1873 2 sink [sink$w_buflen] = .sink [sink$w_buflen] - 2; ! Remove trailing CR/LF
1310 1874 2
1311 1875 2 opcom_buf [opc$b_ms_type] = opc$rq_rqst; ! REQUEST to be sent to operator
1312 1876 2 opcom_buf [opc$b_ms_target] = opc$m_nm_network; ! Sent to network operators
1313 1877 2 opcom_buf [opc$w_ms_status] = 0; ! No status
1314 1878 2 opcom_buf [opc$l_ms_rqstid] = 0; ! No request ID
1315 1879 2
1316 1880 2 opcom_desc [0] = .sink [sink$w_buflen] + $BYTEOFFSET(opc$l_ms_text);
1317 1881 2 opcom_desc [1] = opcom_buf;
1318 1882 2
1319 1883 2 status = $SENDOPR(MSGBUF = opcom_desc); ! Send message to OPCOM
1320 1884 2
1321 1885 2 IF NOT .status ! If error detected,
1322 1886 2 THEN
1323 1887 2 SIGNAL(msg(brdcst), 1, %ASCII 'OPCOM', .status); ! then signal the error
1324 1888 2
```

: 1325

1889 1 END:

				.PSECT \$SPLITS,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		SYSSNDOPR,		EVL\$		BRDCST				
				.PSECT		\$CODE\$,NOWRT,2								
				0004		00000		OUTPUT_OP		COM:				
										.WORD	Save R2	:	1832	
										MOVAB	-2064(SP), SP	:	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	
						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, SYSSNDOPR	:		
	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	00	00000000G					8F	DD	00055	PUSHL	#EVL\$ BRDCST	:		
							04	FB	0005B	CALLS	#4, LIB\$SIGNAL	:		
							04	00062	1\$:	RET		:	1889	

: Routine Size: 99 bytes, Routine Base: \$CODE\$ + 0913

```

1327 1890 1 ROUTINE output_monitor (sink, event, event_length): NOVALUE =
1328 1891 1
1329 1892 1 ---
1330 1893 1
1331 1894 1     Output an event record to a monitor process.
1332 1895 1
1333 1896 1 Inputs:
1334 1897 1
1335 1898 1     sink = Address of sink descriptor block
1336 1899 1     event = Address of event record
1337 1900 1     event_length = Length of event record
1338 1901 1
1339 1902 1 Outputs:
1340 1903 1
1341 1904 1     None
1342 1905 1 ---
1343 1906 1
1344 1907 2 BEGIN
1345 1908 2
1346 1909 2 MAP
1347 1910 2     sink:      REF BBLOCK,      ! Address of sink block
1348 1911 2     event:     REF BBLOCK;      ! Address of event record
1349 1912 2
1350 1913 2 LOCAL
1351 1914 2     status;
1352 1915 2
1353 1916 2
1354 1917 2     ! If a monitor name is specified, then send the event to the remote task.
1355 1918 2     ! We must check for non-null monitor name, since it is valid to have the
1356 1919 2     ! monitor sink turned on without any name, meaning only OPCOM events.
1357 1920 2
1358 1921 2
1359 1922 2 IF .sink [sink$v_error]      ! If in error state,
1360 1923 2 THEN
1361 1924 2     RETURN;                  ! then don't do anything
1362 1925 2
1363 1926 2 IF .sink [sink$b_namelen] NEQ 0 ! If monitor name specified,
1364 1927 2 THEN
1365 1928 2     BEGIN
1366 1929 2     IF .sink [sink$l_closetrn] EQL 0 ! If sink is not yet initialized
1367 1930 2     THEN
1368 1931 2         open_monitor(.sink);      ! then connect to process
1369 1932 2
1370 1933 2     status = $QIOW(FUNC = IOS WRITEVBLK, ! Send event to monitor process
1371 1934 2         CHAN = .sink [sink$w_channel],
1372 1935 2         EFN = evl$c_synch_efn,
1373 1936 2         IOSB = sink [sink$w_iosb],
1374 1937 2         P1 = .event,              ! Address of event record
1375 1938 2         P2 = .event_length);      ! Length of event record
1376 1939 2
1377 1940 2     IF .status                  ! If successfully submitted,
1378 1941 2     THEN
1379 1942 2         status = .sink [sink$w_iosb]; ! then get final I/O status
1380 1943 2
1381 1944 2     IF NOT .status             ! If error detected,
1382 1945 2     THEN
1383 1946 2     BEGIN

```

```

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

```

.EXTRN EVL\$WRITEMON

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

RECEIVER
V04-000

6 2
16-Sep-1984 01:37:57
14-Sep-1984 12:28:54

VAX-11 Bliss-32 V4.0-742 Page 53
DISK\$VMMASTER:[EVL.SRC]RECEIVER.B32;1 (21)

WOF
V04

; Routine Size: 115 bytes. Routine Base: \$CODES + 0976

.....

```

: 1401 1963 1 ROUTINE save_lines (sink, desc, short_form): NOVALUE =
: 1402 1964 1
: 1403 1965 1 |---
: 1404 1966 1 |
: 1405 1967 1 | This is an action routine called by the event formatting
: 1406 1968 1 | routines for each line of output. It appends each line
: 1407 1969 1 | into an buffer associated with the sink followed by a CR/LF.
: 1408 1970 1 |
: 1409 1971 1 | Inputs:
: 1410 1972 1 |
: 1411 1973 1 | sink = Address of sink block
: 1412 1974 1 | desc = Address of descriptor of formatted line
: 1413 1975 1 | short_form = Length of line if short form is desired
: 1414 1976 1 | (-1 if no short form possible)
: 1415 1977 1 |
: 1416 1978 1 | Outputs:
: 1417 1979 1 |
: 1418 1980 1 | The line is appended to the buffer
: 1419 1981 1 |---
: 1420 1982 1
: 1421 1983 2 BEGIN
: 1422 1984 2
: 1423 1985 2 MAP
: 1424 1986 2 sink: REF BBLOCK, ! Address of sink control block
: 1425 1987 2 desc: REF VECTOR; ! Descriptor of output line
: 1426 1988 2
: 1427 1989 2 LOCAL
: 1428 1990 2 linedesc: VECTOR [2], ! Descriptor of the line
: 1429 1991 2 outdesc: VECTOR [2]; ! Output buffer descriptor
: 1430 1992 2
: 1431 1993 2 outdesc [0] = .sink [sink$w_maxbufsiz] - .sink [sink$w_buflen];
: 1432 1994 2 outdesc [1] = .sink [sink$l_buffer] + .sink [sink$w_buflen];
: 1433 1995 2
: 1434 1996 2 linedesc [0] = .desc [0]; ! Copy descriptor locally
: 1435 1997 2 linedesc [1] = .desc [1];
: 1436 1998 2
: 1437 1999 2 IF .short_form GTR 0 ! If there is a short form,
: 1438 2000 2 THEN
: 1439 2001 2 linedesc [0] = .short_form ! then use that because OPCOM
: 1440 2002 2 ELSE ! is limited in output size
: 1441 2003 2 IF .short_form EQL 0 ! If short form means no output,
: 1442 2004 2 THEN
: 1443 2005 2 RETURN; ! then don't append anything at all
: 1444 2006 2
: 1445 P 2007 2 $FAO(%ASCID '!AS!/', ! Append the string to the buffer
: 1446 P 2008 2 outdesc,outdesc, ! Result buffer/length
: 1447 2009 2 linedesc); ! Parameter is descriptor of line
: 1448 2010 2
: 1449 2011 2 sink [sink$w_buflen] = .sink [sink$w_buflen] + .outdesc [0]; ! Update length
: 1450 2012 2
: 1451 2013 1 END;

```

.PSECT \$SPLITS,NOWRT,NOEXE,2

00 00 00 2F 21 53 41 21 000AB P.AAW: .ASCII \!AS!\<0><0><0>

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

.EXTRN SYSS\$FAO
.PSECT \$CODE\$,NOWRT,2

			0004	00000	SAVE_LINES:			
	5E		0C	C2	00002	.WORD	Save R2	
	52	04	AC	D0	00005	SUBL2	#12, SP	
	50	16	A2	3C	00009	MOVL	SINK, R2	
	51	18	A2	3C	C000D	MOVZWL	22(R2), R0	
7E	50		51	C3	00011	MOVZWL	24(R2), R1	
	50	18	A2	3C	00015	SUBL3	R1, R0, OUTDESC	
	04	AE	1A	B240	9E	00019	MOVAB	@26(R2)[R0], OUTDESC+4
	50	08	AC	D0	0001F	MOVL	DESC, R0	
	08	AE	60	7D	00023	MOVQ	(R0), LINEDESC	
	50		0C	AC	D0	00027	MOVL	SHORT_FORM R0
			06	15	0002B	BLEQ	1\$	
	08	AE	50	D0	0002D	MOVL	R0, LINEDESC	
			02	11	00031	BRB	2\$	
			18	13	00033	BEQL	3\$	
		08	AE	9F	00035	PUSHAB	LINEDESC	
		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, SYSS\$FAO	
	18	A2	6E	A0	00049	ADDW2	OUTDESC, 24(R2)	
			04	0004D	3\$:	RET		

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

```
1453 2014 1 ROUTINE open_monitor (sink): NOVALUE =
1454 2015 1
1455 2016 1 ---
1456 2017 1
1457 2018 1       Open the monitor process sink.
1458 2019 1
1459 2020 1       Inputs:
1460 2021 1
1461 2022 1       sink = Address of sink control block
1462 2023 1
1463 2024 1       Outputs:
1464 2025 1
1465 2026 1       None
1466 2027 1 ---
1467 2028 1
1468 2029 2 BEGIN
1469 2030 2
1470 2031 2 MAP
1471 2032 2     sink:          REF BBLOCK;          ! Address of sink block
1472 2033 2
1473 2034 2 LOCAL
1474 2035 2     status,
1475 2036 2     ncb_desc: VECTOR [2],                ! Descriptor of NCB for process
1476 2037 2     ncb:          VECTOR [128,BYTE];      ! NCB
1477 2038 2
1478 P 2039 2     status = $ASSIGN(DEVNAM = %ASCII ' _NET:',
1479 2040 2                 CHAN = sink [sink$w_channel]);
1480 2041 2
1481 2042 2     IF NOT .status                ' If error detected,
1482 2043 2     THEN
1483 2044 2         (SIGNAL(msg(netasn), 0, .status); RETURN); ! then report error
1484 2045 2
1485 2046 2     CH$COPY(CH$RCHAR(evl$gt_localnode+2), evl$gt_localnode+3,
1486 2047 2         8, UPLIT('::TASK='),
1487 2048 2         .sink [sink$b_namelen], sink [sink$t_name],
1488 2049 2         1, UPLIT(''),
1489 2050 2         0, 128, ncb);
1490 2051 2
1491 2052 2     ncb_desc [0] = CH$RCHAR(evl$gt_localnode+2)+8+.sink [sink$b_namelen]+1;
1492 2053 2     ncb_desc [1] = ncb;
1493 2054 2
1494 P 2055 2     status = $QIOW(FUNC = IOS_ACCESS,      ! Connect to monitor process
1495 PP 2056 2                 CHAN = .sink [sink$w_channel],
1496 PP 2057 2                 EFN = evl$sc_synch_efn,
1497 P 2058 2                 IOSB = sink [sink$w_iosb],
1498 2059 2                 P2 = ncb_desc);
1499 2060 2
1500 2061 2     IF .status                    ! If successfully submitted,
1501 2062 2     THEN
1502 2063 2         status = .sink [sink$w_iosb];    ! then get final I/O status
1503 2064 2
1504 2065 2     IF NOT .status                ! If error detected,
1505 2066 2     THEN
1506 2067 2         (SIGNAL(msg(openmon), 1, ncb_desc, .status); RETURN);
1507 2068 2
1508 2069 2     sink [sink$l_closetrn] = close_monitor; ! Set connection established
1509 2070 2
```

```

: 1510      2071 2 IF .evl$gl_logmask [elg$v_monopn]      ! If logging monitor connections,
: 1511      2072 2 THEN
: 1512      2073 2     SIGNAL(msg(logopnm), 3, .ncb_desc [0], .ncb_desc [1], 0);
: 1513      2074 2
: 1514      2075 1 END;

```

```

.PSECT $SPLITS,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:
5E FF78 CE 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
7E 7C 00007 MOVAB -136(SP), SP
56 04 AC D0 00009 CLRQ -(SP)
1E A6 9F 0000D MOVL SINK, R6
00000000' CF 9F 00010 PUSHAB 30(R6)
00 04 FB 00014 PUSHAB P.AAX
5B 50 D0 0001B CALLS #4, SYSS$ASSIGN
12 5B E8 0001E MOVL R0, STATUS
5B DD 00021 BLBS STATUS, 1$
7E D4 00023 PUSHL STATUS
00000000G 00 00000000G 8F DD 00025 PUSHL #EVL$_NETASN
03 FB 0002B CALLS #3, LIB$SIGNAL
04 00032 RET
5A 0000G CF 9A 00033 1$: MOVZBL EVL$GT_LOCALNODE+2, R10
59 2E A6 9A 00038 MOVZBL 46(R6), R9
58 80 8F 9A 0003C MOVZBL #128, R8
57 6E 9E 00040 MOVAB NCB, R7
58 00 0000G CF 5A 2C 00043 MOVCS 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 MOVCS #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 MOVCS 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 MOVCS #1, P.ABA, #0, R8, (R7)
67 00079
50 0000G CF 9A 0007A 2$: MOVZBL EVL$GT_LOCALNODE+2, R0

```

		51	2E	A6	9A	0007F	MOVZBL	46(R6), R1	
		50		51	CO	00083	ADDL2	R1, R0	
	FB	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, SYSSQIOW	
		5B		50	DD	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	PUSHL	STATUS	2067
			F8	AD	9F	000BB	PUSHAB	NCB_DESC	
				01	DD	000BE	PUSHL	#1	
				8F	DD	000C0	PUSHL	#EVL\$ OPENMON	
00000000G		00	00000000G	04	FB	000C6	CALLS	#4, LIB\$SIGNAL	
				04	04	000CD	RET		
	22	A6	0000V	CF	9E	000CE	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	
				8F	DD	000E2	PUSHL	#EVL\$ LOGOPNM	
00000000G		00	00000000G	05	FB	000E8	CALLS	#5, LIB\$SIGNAL	
				04	04	000EF	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_closetrn] = 0;          ! Mark sink closed
: 1550 2110 2
: 1551 2111 1 END;

```

```

                                .EXTRN  EVL$_CLOSEMON
                                0004 0000 CLOSE_MONITOR:
                                .WORD    Save R2                                : 2076
                                SUBL2    #4, SP
                                MOVL     SINK, R2                                : 2100
                                MOVZBL   46(R2), NCB_DESC
                                MOVAB    47(R2), NCB_DESC+4
                                MOVZWL   30(R2), -(SP)
                                CALLS    #1, SYSSDASSGN
                                BLBS     STATUS, 1$
                                PUSHL    STATUS
                                PUSHAB   NCB_DESC
                                PUSHL    #1
                                PUSHL    #EVL$_CLOSEMON
                                CALLS    #4, LIB$SIGNAL
                                CLRL     34(R2)
                                RET
                                04 00037

```

RECEIVER
V04-000

J 2
16-Sep-1984 01:37:57
14-Sep-1984 12:28:54

VAX-11 Bliss-32 V4.0-742 Page 60
DISK\$VMSMASTER:[EVL.SRC]RECEIVER.B32;1 (24)

WOF
V04

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

; F

RECEIVER
V04-000

K 2
16-Sep-1984 01:37:57
14-Sep-1984 12:28:54

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EVL.SRC]RECEIVER.B32;1 (25) Page 61

WOR
V04

: 1553 2112 1 END
: 1554 2113 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	92	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$GLOBALS	1	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$SPLITS	212	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODES	2911	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing 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=(ENP\$: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

The image displays a grid of 100 small terminal window screenshots, arranged in 10 rows and 10 columns. Each window shows a different VAX/VMS command and its output. The windows are arranged in a grid. Many windows have titles like 'LPMULT B32', 'DRMAST MAR', 'ADDRIVER MAR', 'TORIVER MAR', 'USSTEST MAR', 'GBLSECURF MAR', 'USSDISP MAR', 'XADDRIVER MAR', 'LBRMAC MAR', 'LABLOCIN MAR', 'DTE_DF03 MAR', 'DRSLU MAR', 'SECRET MAR', 'WORKO LIS', and 'EXAMPLES'. Each window contains text-based data, including system status, command prompts, and output results.