



```

LL      IIIIII  NN      NN  KK      KK      SSSSSSSS  UU      UU  BBBB8888  SSSSSSSS
LL      IIIIII  NN      NN  KK      KK      SSSSSSSS  UU      UU  BBBB8888  SSSSSSSS
LL      II      NN      NN  KK      KK      SS        UU      UU  BB      BP  SS
LL      II      NN      NN  KK      KK      SS        UU      UU  BB      dB  SS
LL      II      NNNN    NN  KK      KK      SS        UU      UU  BB      BB  SS
LL      II      NNNN    NN  KK      KK      SS        UU      UU  BB      BB  SS
LL      II      NN  NN  NN  KKKKKK  KK      SS        UU      UU  BB      BB  SS
LL      II      NN  NN  NN  KKKKKK  KK      SS        UU      UU  BB      BB  SS
LL      II      NN      NNNN  KK      KK      SS        UU      UU  BB      BB  SS
LL      II      NN      NNNN  KK      KK      SS        UU      UU  BB      BB  SS
LL      II      NN      NN  KK      KK      SS        UU      UU  BB      BB  SS
LL      II      NN      NN  KK      KK      SS        UU      UU  BB      BB  SS
LLLLLLLLLLLL  IIIIII  NN      NN  KK      KK      SSSSSSSS  UUUUUUUUUU  BBBB8888  SSSSSSSS
LLLLLLLLLLLL  IIIIII  NN      NN  KK      KK      SSSSSSSS  UUUUUUUUUU  BBBB8888  SSSSSSSS

```

```

LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLLLL  IIIIII  SSSSSSSS

```

L  
V  
.....

```

1 0001 0 %title 'LINKSUBS - Phone Link Subroutines'
2 0002 0      module linksubs (
3 0003 1      ident='V04-000') = begin
4 0004 1
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
9 0009 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
10 0010 1 *  ALL RIGHTS RESERVED. *
11 0011 1 *
12 0012 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
13 0013 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
14 0014 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
15 0015 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
16 0016 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
17 0017 1 *  TRANSFERRED. *
18 0018 1 *
19 0019 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
20 0020 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
21 0021 1 *  CORPORATION. *
22 0022 1 *
23 0023 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
24 0024 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
25 0025 1 *
26 0026 1 *****
27 0027 1
28 0028 1
29 0029 1
30 0030 1 **
31 0031 1 Facility:      VAX/VMS Telephone Facility, Phone Link Subroutines
32 0032 1
33 0033 1 Abstract:      This module contains the subroutines necessary to support
34 0034 1 the establishment and use of phone links, both local
35 0035 1 and remote.
36 0036 1
37 0037 1
38 0038 1 Environment:
39 0039 1
40 0040 1 Author: Paul C. Anagnostopoulos, Creation Date: 17 November 1980
41 0041 1
42 0042 1 Modified By:
43 0043 1
44 0044 1 V03-006 BLS0251      Benn Schreiber      8-Dec-1983
45 0045 1 Modify mailbox name so that all numeric usernames
46 0046 1 do not look like cluster device names. Also allow
47 0047 1 $$$_IVDEVNAM from $ASSIGN. Convert to use $TRNLNM.
48 0048 1
49 0049 1 V03-005 PCA1020      Paul C. Anagnostopoulos 27-May-1983
50 0050 1 Add check for shared memory mailboxes and signal fatal
51 0051 1 error if present.
52 0052 1
53 0053 1
54 0054 1 V03-004 MHB0088      Mark Bramhall      8-Feb-1983
55 0055 1 Corrected underscore handling in PHN$ESTAB_LINK.
56 0056 1
57 0057 1 V03-003 MHB0087      Mark Bramhall      17-Jan-1983

```

LINKSUBS  
V04-000

LINKSUBS - Phone Link Subroutines

G 3  
16-Sep-1984 02:11:44 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:53:27 [PHONE.SRC]LINKSUBS.B32;1

Page 2  
(1)

```

: 58      0058 1  |
: 59      0059 1  |
: 60      0060 1  |
: 61      0061 1  |
: 62      0062 1  |
: 63      0063 1  |
: 64      0064 1  |
: 65      0065 1  |
: 66      0066 1  |
: 67      0067 1  |
: 68      0068 1  |--

```

Added "pass through" routing capability.  
PHN\$ESTAB LINK now always displays its status via  
PHN\$INFORM; its callers just check the returned status.

V03-002 PCA1004 Paul C. Anagnostopoulos 8-Nov-1982  
\$ASSIGN no longer returns SSS\_IVDEVNAM when assigning to  
a mailbox that doesn't exist. It now returns SSS\_NOSUCHDEV.

V03-001 PCA0041 Paul Anagnostopoulos 26-Mar-1982  
Major changes to convert from process name to user name.

```
.. 70 0069 1 %sbttl 'Module Declarations'
.. 71 0070 1
.. 72 0071 1   Libraries and Requires:
.. 73 0072 1
.. 74 0073 1
.. 75 0074 1 library 'sys$library:starlet.l32';
.. 76 0075 1 require 'phonereq';
.. 77 0404 1
.. 78 0405 1
.. 79 0406 1   Table of Contents:
.. 80 0407 1
.. 81 0408 1
.. 82 0409 1 forward routine
.. 83 0410 1     phn$mbx_enable: novalue,
.. 84 0411 1     phn$estab_link,
.. 85 0412 1     phn$mbx_name: novalue,
.. 86 0413 1     phn$break_link: novalue,
.. 87 0414 1     phn$break_call: novalue,
.. 88 0415 1     phn$send_smb: novalue,
.. 89 0416 1     phn$orce_links: novalue,
.. 90 0417 1     phn$forced_link: novalue;
.. 91 0418 1
.. 92 0419 1
.. 93 0420 1   External References:
.. 94 0421 1
.. 95 0422 1
.. 96 0423 1 external routine
.. 97 0424 1     phn$term_characteristic,
.. 98 0425 1     phn$cmp_target,
.. 99 0426 1     phn$fresh_screen,
100 0427 1     phn$inform,
101 0428 1     phn$kill_pub,
102 0429 1     phn$make_pub,
103 0430 1     phn$make_tsb,
104 0431 1     phn$queue_smb,
105 0432 1     phn$read_slave,
106 0433 1     uns$net_connect: addressing_mode(general);
107 0434 1
108 0435 1
109 0436 1   Own Variables:
110 0437 1
111 0438 1 bind
112 0439 1     lnm_process_desc = $descriptor('LNMS$PROCESS'),
113 0440 1     lnm_job_desc = $descriptor('LNMS$JOB'),
.. 114 0441 1     slave_task_desc = $descriptor('PHN$SLAVE_TASK_SPECIFIER');
```

```
116 0442 1 %sbttl 'PHN$MBX_ENABLE - Enable Mailbox ASTs'
117 0443 1 ++
118 0444 1 Functional Description:
119 0445 1 This routine is called to enable an AST for our receive mailbox.
120 0446 1 This allows us to be notified when some other process puts a
121 0447 1 steering message into our mailbox. The AST itself just creates
122 0448 1 a standard steering message block from the message. The message
123 0449 1 has the following three parts:
124 0450 1
125 0451 1 1. The 1-byte message type code.
126 0452 1
127 0453 1 2. The complete node/user name string of the sender,
128 0454 1 followed by an eofrom character.
129 0455 1
130 0456 1 3. Any additional message text, if required.
131 0457 1
132 0458 1 Formal Parameters:
133 0459 1 none
134 0460 1
135 0461 1 Implicit Inputs:
136 0462 1 global data
137 0463 1
138 0464 1 Implicit Outputs:
139 0465 1 global data
140 0466 1
141 0467 1 Returned Value:
142 0468 1 none
143 0469 1
144 0470 1 Side Effects:
145 0471 1
146 0472 1 --
147 0473 1
148 0474 1
149 0475 2 global routine phn$mbx_enable: novalue = begin
150 0476 2
151 0477 2 own
152 0478 2 mbx_iosb: block[8,byte],
153 0479 2 mbx_buf: vector[phn$k_mbxsize,byte];
154 0480 2
155 0481 2 local
156 0482 2 status: long,
157 0483 2 op: ref pub; ! Pointer to our PUB.
158 0484 2
159 0485 2
160 0486 2 ! This internal routine is the AST handler. We have to build a standard
161 0487 2 ! steering message block from the mailbox data.
162 0488 2
163 0489 2 routine mbx_ast: novalue = begin
164 0490 2
165 0491 2 local
166 0492 2 status: long,
167 0493 2 mbx_buf_dsc: descriptor;
168 0494 2
169 0495 2 ! First we have to check the IOSB from the mailbox read.
170 0496 2
171 0497 2 check (.mbx_iosb[0,0,16,0]);
172 0498 2
```

```

: 173      0499 3 ! Now we build a descriptor for the message text portion of the steering
: 174      0500 3 ! message. We can then queue an SMB with the message type and text.
: 175      0501 3
: 176      0502 3 mbx_buf_dsc[len] = .mbx_iosb[2,0,16,0] - 1;
: 177      0503 3 mbx_buf_dsc[ptr] = mbx_buf[1];
: 178      0504 3 phn$queue_smb(.mbx_buf[0],mbx_buf_dsc);
: 179      0505 3
: 180      0506 3 ! Finally, we have to enable another read from our mailbox.
: 181      0507 3
: 182      0508 3 phn$mbx_enable();
: 183      0509 3 return;
: 184      0510 3
: 185      0511 2 end;

```

```

.TITLE LINKSUBS LINKSUBS - Phone Link Subroutines
.IDENT \V04-000\

```

```

.PSECT $PLITS$,NOWRT,NOEXE,2

```

53	53	45	43	4F	52	50	24	4D	4E	4C	00000	P.AAB:	.ASCII	\LNMS\$PROCESS\	:					
											0000B		.BLKB	1	:					
										0000000B	0000C	P.AAA:	.LONG	11	:					
										00000000'	00010		.ADDRESS	P.AAB	:					
				42	4F	4A	24	4D	4E	4C	00014	P.AAD:	.ASCII	\LNMS\$JOB\	:					
											0001B		.BLKB	1	:					
										00000007	0001C	P.AAC:	.LONG	7	:					
										00000000'	00020		.ADDRESS	P.AAD	:					
SF	4B	53	41	54	SF	45	56	41	4C	53	24	4E	48	50	00024	P.AAF:	.ASCII	\PHN\$SLAVE_TASK_SPECIFIER\	:	
						52	45	49	46	49	43	45	50	53	00033					:
												00000018	0003C	P.AAE:	.LONG	24				:
											00000000'	00040		.ADDRESS	P.AAF					:

```

.PSECT $OWNS$,NOEXE,2

```

```

00000 MBX_IOSB:
      .BLKB 8
00008 MBX_BUF: .BLKB 256

```

```

LNMS_PROCESS_DESC= P.AAA
LNMS_JOB_DESC= P.AAC
SLAVE_TASK_DESC= P.AAE

```

```

.EXTRN PHN$ OK, PHN$ ANSWERED
.EXTRN PHN$ BUSYCALL, PHN$ CANCELL
.EXTRN PHN$ CANTREACH, PHN$ CONFCALL
.EXTRN PHN$ DEAD, PHN$ DECNELINK
.EXTRN PHN$ DIRCAN, PHN$ FACSCAN
.EXTRN PHN$ HELPCAN, PHN$ HUNGUP
.EXTRN PHN$ JUSTRANG, PHN$ LOGGEDOFF
.EXTRN PHN$ REJECTED, PHN$ RING
.EXTRN PHN$ REJECTJUNK
.EXTRN PHN$ SENDINGMAIL
.EXTRN PHN$ BADCMD, PHN$ BADHELP
.EXTRN PHN$ BADMAILCMD
.EXTRN PHN$ BADSMB, PHN$ BADSPEC
.EXTRN PHN$ HELPMISSING
.EXTRN PHN$ IVREDUNANS

```

```
.EXTRN PHNS_IVREDUNCALL
.EXTRN PHNS_LINKERROR, PHNS_NEEDUSER
.EXTRN PHNS_NOCALL, PHNS_NOHOLDS
.EXTRN PHNS_NOPORTS, PHNS_NOPRIV
.EXTRN PHNS_NOPROC, PHNS_NOTCONV
.EXTRN PHNS_ONLYNODE, PHNS_PHONEBUSY
.EXTRN PHNS_REMOTEERROR
.EXTRN PHNS_TARGTERM, PHNS_UNPLUGGED
.EXTRN PHNS_BADTERM, PHNS_SHAREDMBX
.EXTRN PHNS_INPUTTERM, PHNSGQ_NODE_NAME
.EXTRN PHNSGQ_SWITCH_HOOK
.EXTRN PHNSGL_VIEWPORT_SIZE
.EXTRN PHNSGB_SCROLL, PHNSGQ_PUBHEAD
.EXTRN PHNSGB_FLAGS, PHNSTERM_CHARACTERISTIC
.EXTRN PHNSCMP_TARGET, PHNSFRESH_SCREEN
.EXTRN PHNSINFORM, PHNSKILL_PUB
.EXTRN PHNSMAKE_PUB, PHNSMAKE_TSB
.EXTRN PHNSQUEUE_SMB, PHNSREAD_SLAVE
.EXTRN UNS$NET_CONNECT
```

.PSECT \$CODE\$,NOWRT,2

			0004	00000	MBX_AST:	.WORD	Save R2	:	0489	
	52	0000'	CF	9E	00002	MOVAB	MBX_IOSB, R2	:		
	5E		08	C2	00007	SUBL2	#8, -SP	:		
	0A		62	E8	0000A	BLBS	MBX_IOSB, 1\$	:	0497	
	7E		62	3C	0000D	MOVZWL	MBX_IOSB, -(SP)	:		
6E	00000000G	00	01	FB	00010	CALLS	#1, LIB\$SIGNAL	:		
	02	A2	01	A3	00017	1\$: SUBW3	#1, MBX_IOSB+2, MBX_BUF_DSC	:	0502	
	04	AE	09	A2	9E	0001C	MOVAB	MBX_BUF+1, MBX_BUF_DSC+4	:	0503
			5E	DD	00021	PUSHL	SP	:	0504	
		7E	08	A2	9A	00023	MOVZBL	MBX_BUF, -(SP)	:	
	0000G	CF	02	FB	00027	CALLS	#2, PHNS\$QUEUE_SMB	:		
	0000V	CF	00	FB	0002C	CALLS	#0, PHNS\$MBX_ENABLE	:	0508	
			04	00	00031	RET		:	0511	

; Routine Size: 50 bytes, Routine Base: \$CODE\$ + 0000



```

: 187      0512 2 ! To actually enable a read from our mailbox, we just do the appropriate
: 188      0513 2 ! mailbox $qio.
: 189      0514 2
: 190      0515 2 op = .phn$gg_pubhead[0];
: 191      P 0516 2 status = $qio(efn=phn$k_ourmbxefn,
: 192      P 0517 2          chan=.op[pub_w_channel],
: 193      P 0518 2          func=io$_readvblk,
: 194      P 0519 2          iosb=mbx_iosb,
: 195      P 0520 2          astadr=mbx_ast,
: 196      P 0521 2          p1=mbx_buf,
: 197      0522 2          p2=phn$k_mbxsize);
: 198      0523 2 check (.status);
: 199      0524 2 return;
: 200      0525 2
: 201      0526 1 end;

```

					.EXTRN	SYSSQIO	
			0000	00000	.ENTRY	PHN\$MBX_ENABLE, Save nothing	: 0475
	50	0000G	CF	D0 00002	MOVL	PHN\$GQ_PUBHEAD, OP	: 0515
			7E	7C 00007	CLRQ	-(SP)	: 0522
			7E	7C 00009	CLRQ	-(SP)	:
	7E	0100	8F	3C 0000B	MOVZWL	#256, -(SP)	:
		0000'	CF	9F 00010	PUSHAB	MBX_BUF	:
			7E	D4 00014	CLRL	-(SP)	:
		B5	AF	9F 00016	PUSHAB	MBX_AST	:
		0000'	CF	9F 00019	PUSHAB	MBX_IOSB	:
			31	DD 0001D	PUSHL	#49	:
	7E	00F4	C0	3C 0001F	MOVZWL	244(OP), -(SP)	:
			05	DD 00024	PUSHL	#5	:
	00000000G	00	0C	FB 00026	CALLS	#12, SYSSQIO	:
		09	50	E8 0002D	BLBS	STATUS, 1\$	: 0523
			50	DD 00030	PUSHL	STATUS	:
	00000000G	00	01	FB 00032	CALLS	#1, LIB\$SIGNAL	:
			04	00039 1\$:	RET		: 0526

; Routine Size: 58 bytes, Routine Base: \$CODE\$ + 0032

```
203 0527 1 %sbttl 'PHN$ESTAB_LINK - Establish a Link'
204 0528 1 ++
205 0529 1 Functional Description:
206 0530 1 This routine is called to establish a link between us and some other
207 0531 1 place in the world. There are four possible cases:
208 0532 1 1. A link between us and ourselves.
209 0533 1 2. A link between us and some other local user.
210 0534 1 3. A link between us and a remote node (for information).
211 0535 1 4. A link between us and some other remote user.
212 0536 1
213 0537 1 Formal Parameters:
214 0538 1 target_spec Address of descriptor of complete node/user name spec
215 0539 1 of the target.
216 0540 1 pub_address Address of longword in which to return address of PUB
217 0541 1 describing established link. The PUB is marked
218 0542 1 temporary.
219 0543 1
220 0544 1 Implicit Inputs:
221 0545 1 global data
222 0546 1
223 0547 1 Implicit Outputs:
224 0548 1 global data
225 0549 1
226 0550 1 Returned Value:
227 0551 1 phn$_badspec Target spec syntax was invalid.
228 0552 1 phn$_cantreach Cannot reach the target right now.
229 0553 1 phn$_needuser Tried to establish link to home node w/o user name
230 0554 1 phn$_nopriv Do not have the necessary privileges.
231 0555 1 phn$_noproc No process owned by the user is available.
232 0556 1 phn$_remoteerror Some sort of error during remote I/O.
233 0557 1 phn$_targterm None of the user's terminals are usable by PHONE.
234 0558 1 DECnet status Problem with remote link.
235 0559 1
236 0560 1 Side Effects:
237 0561 1 Any error is displayed via PHN$INFORM; all callers should only be
238 0562 1 checking the returned status.
239 0563 1
240 0564 1 --
241 0565 1
242 0566 1
243 0567 2 global routine phn$estab_link(target_spec, pub_address) = begin
244 0568 2
245 0569 2 own
246 0570 2 path_error_done: byte;
247 0571 2
248 0572 3 routine path_error(error_code, fao_count, fao_text): novalue = begin
249 0573 3 builtin
250 0574 3 nullparameter;
251 0575 4 if not .path_error_done then (
252 0576 4 path_error_done = true;
253 0577 4 if not nul(parameter(3)) then
254 0578 4 phn$inform(.error_code, .fao_text)
255 0579 4 else
256 0580 4 phn$inform(.error_code);
257 0581 3 );
258 0582 2 end;
```

.PSECT \$OWNS\$,NOEXE,2

00108 PATH\_ERROR\_DONE:  
.B[KB] 1

.PSECT \$CODE\$,NOWRT,2

```

                                0000 00000 PATH_ERROR:
                                .WORD Save nothing
0000' 23 0000' CF E8 00002          BLBS PATH_ERROR_DONE, 2$
                                MOV# #1, PATH_ERROR_DONE
                                CF      01 90 00007
                                CMPB (AP), #3
                                03      6C 91 0000C
                                BLSSU 1$
                                OC      11 1F 0000F
                                TSTL 12(AP)
                                OC      0C AC D5 00011
                                BEQL 1$
                                OC      0C AC DD 00014
                                PUSH# #1, PATH_ERROR_DONE
                                04      AC DD 00016
                                PUSH# #1, PATH_ERROR_DONE
                                0000G CF 04 AC DD 00019
                                CALLS #2, PRN$INFORM
                                02      FB 0001C
                                RET
                                04      AC DD 00021
                                0000G CF 04 AC DD 00022 1$:
                                PUSH# #1, PATH_ERROR_DONE
                                01      FB 00025
                                CALLS #1, PRN$INFORM
                                04      0002A 2$:
                                RET
    
```

: Routine Size: 43 bytes, Routine Base: \$CODE\$ + 006C

```

259 0583 2
260 0584 2 bind
261 0585 2 target_spec_dsc = .target_spec: descriptor;
262 0586 2
263 0587 2 own
264 0588 2 own_described_buffer(user_name,12),
265 0589 2 parent_pid: long,
266 0590 2 own_described_buffer(term_number,7),
267 0591 2 path_list: block[dsc$k_d_bln, byte] field(descriptor_fields)
268 0592 2 preset([dsc$w_length] = 0, [dsc$b_dtype] = dsc$k_dtype_t,
269 0593 2 [dsc$b_class] = dsc$k_class_d, [dsc$a_pointer] = 0);
270 0594 2
271 0595 2 bind
272 0596 2 get_proc = uplit(word(12),word(jpi$username),
273 0597 2 long(user_name+8),
274 0598 2 long(user_name),
275 0599 2 word(4),word(jpi$owner),
276 0600 2 long(parent_pid),
277 0601 2 long(0),
278 0602 2 word(7),word(jpi$terminal),
279 0603 2 long(term_number+8),
280 0604 2 long(term_number),
281 0605 2 long(0));
282 0606 2
283 0607 2 local
284 0608 2 status: long,
285 0609 2 tp: ref pub, ! Pointer to new target PUB.
    
```

```
.. 286      0610 2      op: ref pub,                ! Pointer to our PUB.
.. 287      0611 2      wild_pid: long,
.. 288      0612 2      potential: long, usable: long;
.. 289      0613 2
.. 290      0614 2      . We begin by making a PUB for this link.  If we can't, just return a status.
.. 291      0615 2
.. 292      0616 2      status = phn$make pub(.target_spec,.pub_address);
.. 293      0617 3      if .status eglu phn$ badspec then (
.. 294      0618 3          phn$inform(phn$ badspec);
.. 295      0619 3          return phn$_badspec;
.. 296      0620 3      );
.. 297      0621 2      check (.status);
.. 298      0622 2      tp = ..pub_address;
.. 299      0623 2
.. 300      0624 2      ! Now we split up depending upon whether it is a local or remote link.
.. 301      0625 2
.. 302      0626 3      begin
.. 303      0627 3      bind
.. 304      0628 3          target_tsb = tp[pub_b_tsb]: tsb,
.. 305      0629 3          target_name = target_tsb[tsb_q_tkndsc,..target_tsb[tsb_w_tkncount]]: descriptor;
.. 306      0630 3
.. 307      0631 4      if not .target_tsb[tsb_v_remote] then (
.. 308      0632 4
.. 309      0633 4          ! We have a local link.  Make sure a user name was specified
.. 310      0634 4          ! in the spec.  We can't make a link just to our own node.
.. 311      0635 4
.. 312      0636 5          if not .target_tsb[tsb_v_user] then (
.. 313      0637 5              phn$kill_pub(.tp);
.. 314      0638 5              phn$inform(phn$_needuser);
.. 315      0639 5              return phn$_needuser;
.. 316      0640 4          );
.. 317      0641 4
.. 318      0642 4          ! Now we split up depending upon whether we are linking to ourselves
.. 319      0643 4          ! or another local user.
.. 320      0644 4
.. 321      0645 4          op = .phn$gq_pubhead[0];
.. 322      0646 4          if phn$cmp_target(tp[pub_b_tsb],op[pub_b_tsb]) then
.. 323      0647 4
.. 324      0648 4              ! We are linking to ourselves.  Just get the channel from
.. 325      0649 4              ! our PUB and fill it into the new PUB.  If there isn't one
.. 326      0650 4              ! we didn't have sufficient privilege.
.. 327      0651 4
.. 328      0652 5              if .op[pub_w_channel] nequ 0 then (
.. 329      0653 5                  tp[pub_w_channel] = .op[pub_w_channel];
.. 330      0654 5                  return phn$_ok;
.. 331      0655 5              ) else (
.. 332      0656 5                  phn$kill_pub(.tp);
.. 333      0657 5                  phn$inform(phn$_nopriv);
.. 334      0658 5                  return phn$_nopriv;
.. 335      0659 4              );
```

```

: 337      0660  4      ! We are linking to another local user. We must determine if
: 338      0661  4      ! anyone who is logged in fits the bill.
: 339      0662  4
: 340      0663  4      potential = usable = 0;
: 341      0664  4      wild_pid = -1;
: 342      0665  5      loop (
: 343      0666  5
: 344      0667  5          ! Get information on the next process. If there aren't
: 345      0668  5          ! any more, then we're done.
: 346      0669  5
: 347      P 0670  5          status = $getjpiw(efn=phn$k_getjpiefn,
: 348      P 0671  5              pidadr=wild_pid,
: 349      0672  5              itmlst=get_proc);
: 350      0673  5
: 351      0674  5      exitif (.status eqlu ss$_nomoreproc);
: 352      0675  5
: 353      0676  5          ! If we got a process, then determine if it is a detached
: 354      0677  5          ! interactive process owned by the target.
: 355      0678  5
: 356      0679  6          if .status eqlu ss$_normal then (
: 357      0680  6              if ch$eql(.target_name[len],.target_name[ptr], .user_name[len],.user_name[ptr],' ')
: 358      0681  6                  .parent_pid eqlu 0 and
: 359      0682  6                  .term_number[len] nequ 0 then (
: 360      0683  7                      ! We got a potential candidate. Make sure that
: 361      0684  7                      ! their terminal is usable by PHONE.
: 362      0685  7
: 363      0686  7                      inc (potential);
: 364      0687  7                      if phn$term_characteristic(term_number,tt$m_scope) then
: 365      0688  7                          inc (usable);
: 366      0689  7
: 367      0690  7                      );
: 368      0691  6              );
: 369      0692  5          );
: 370      0693  4      );
: 371      0694  4
: 372      0695  4          ! If there are no potential processes, or there are but no usable
: 373      0696  4          ! terminals, then return an appropriate status and flush the link.
: 374      0697  4
: 375      0698  5          if .potential eqlu 0 then (
: 376      0699  5              phn$kill_pub(.tp);
: 377      0700  5              phn$inform(phn$_noproc);
: 378      0701  5              return phn$_noproc;
: 379      0702  4          );
: 380      0703  5          if .usable eqlu 0 then (
: 381      0704  5              phn$kill_pub(.tp);
: 382      0705  5              phn$inform(phn$_targterm);
: 383      0706  5              return phn$_targterm;
: 384      0707  4          );
: 385      0708  4
: 386      0709  4          ! Now we must create a receive mailbox for the target. We begin by
: 387      0710  4          ! building a name for the mailbox and trying to assign to it, in
: 388      0711  4          ! case someone else has already created it.
: 389      0712  4
: 390      0713  5      begin
: 391      0714  5      local
: 392      0715  5          local_described_buffer(mbx_name,4+32);
: 393      0716  5

```

```

: 394      0717 5      phn$mbx_name(target_name,mbx_name);
: 395      0718 5      status = $assign(devnam=mbx_name,
: 396      0719 5              chan=tp[pub_w_channel]);
: 397      0720 6      if (.status nequ ss$nosuchdev)
: 398      0721 6          and (.status nequ ss$_ivdevnam) then (
: 399      0722 6              check (.status);
: 400      0723 6              return phn$_ok;
: 401      0724 5      );
: 402      0725 5
: 403      0726 5      ! Nope, mailbox doesn't already exist. Create a permanent one with
: 404      0727 5      ! the name and mark it for deletion so we don't leave crud around.
: 405      0728 5
: 406      P 0729 5      status = $crembx(prmflg=1,
: 407      P 0730 5              chan=tp[pub_w_channel],
: 408      P 0731 5              maxmsg=phn$_mbxsize,
: 409      0732 5              lognam=mbx_name);
: 410      0733 6      if .status eqlu ss$_nopriv then (
: 411      0734 6          phn$kill_pub(.tp);
: 412      0735 6          phn$inform(phn$_nopriv);
: 413      0736 6          return phn$_nopriv;
: 414      0737 5      );
: 415      0738 5      check (.status);
: 416      0739 5      status = $delmbx(chan=.tp[pub_w_channel]);
: 417      0740 5      check (.status);
: 418      0741 5      return phn$_ok;
: 419      0742 5
: 420      0743 4      end;
: 421      0744 3 );
```

```
423 0745 3 ! We are to establish a link to a remote node or user. This requires
424 0746 3 ! us to make a logical link to the remote node. The so-called task specifier
425 0747 3 ! is built as follows:
426 0748 3 !     normal case: [node:...].node:''29='
427 0749 3 !     debugging:  {whatever is in PHN$SLAVE_TASK_SPECIFIER}
428 0750 3
429 0751 3 phn$inform(phn$_dechnetlink);
430 0752 3
431 0753 4 begin
432 0754 4
433 0755 4 bind
434 0756 4     whole_target = target_tsb[tsb_q_tkndsc, 0]: descriptor;
435 0757 4
436 0758 4 local
437 0759 4     trnlmlst : $itmlst_decl(items=1),
438 0760 4     local_described_buffer(specifier_buf, nam$c_maxrss);
439 0761 4
440 0762 4 !
441 0763 4 ! Translate PHN$SLAVE_TASK_SPECIFIER (used for debugging only). Look in
442 0764 4 ! the process table and the job table (if not found in process table).
443 0765 4 !
444 P 0766 4 $itmlst_init(itmlst=trnlmlst,
445 P 0767 4     (itmcod=lnm$string,bufadr=.specifier_buf[ptr],
446 P 0768 4     bufsiz=nam$c_maxrss,retlen=specifier_buf));
447 P 0769 4
448 P 0770 4 status = $strlnm(attr=%REF(lnm$m_case_blind),
449 P 0771 4     tabnam=lnm_process_desc,
450 P 0772 4     lognam=slave_task_desc,
451 P 0773 4     itmlst=trnlmlst);
452 0774 4
453 0775 4 if .status nequ ss$_normal
454 0776 5 then begin
455 0777 5     specifier_buf[len] = nam$c_maxrss;
456 P 0778 5     status = $strlnm(attr=%REF(lnm$m_case_blind),
457 P 0779 5     tabnam=lnm_job_desc,
458 P 0780 5     lognam=slave_task_desc,
459 P 0781 5     itmlst=trnlmlst);
460 0782 4
461 0783 4 end;
462 0784 5 if .status nequ ss$_normal then (
463 0785 5     ch$copy(.whole_target[len] - .target_name[len], .whole_target[ptr],
464 0786 5     5, uplit byte(''29=''),
465 0787 5     nam$c_maxrss, .specifier_buf[ptr]);
466 0788 5     specifier_buf[len] = .whole_target[len] - .target_name[len] + 5;
467 0789 4 );
468 0790 4
469 0791 4 ! Now we can actually create the logical link to the remote node. This is
470 0792 4 ! using the UNS$NET_CONNECT routine. It will use the "pass through" protocol
471 0793 4 ! if needed to make the connection. The final routing is also returned.
472 0794 4
473 0795 4 path_error_done = false;
474 0796 4 status = uns$net_connect(specifier_buf, tp[pub_w_channel],
475 0797 4     0, path_list, 0, path_error);
476 0798 5 if not .status then (
477 0799 5     phn$kill_pub(.tp);
478 0800 5     if not .path_error_done then
479 0801 5     phn$inform(.status);
```

```
480 0802 5      return .status;
481 0803 4 );
482 0804 4
483 0805 4 ! Check to see if the routing needs updating.  If so, do so.
484 0806 4
485 0807 4 if not ch$eq(.path_list[.len], .path_list[ptr],
486 0808 4      ;whole_target[.len] - .target_name[.len], .whole_target[ptr],
487 0809 5      ;') then (
488 0810 5      local
489 0811 5      dsc_cnt, path_len, path_ptr;
490 0812 5      ch$move(.target_name[.len], .target_name[ptr], .specifier_buf[ptr]);
491 0813 5      ch$move(.path_list[.len], .path_list[ptr], .whole_target[ptr]);
492 0814 5      ch$move(.target_name[.len], .specifier_buf[ptr],
493 0815 5      .whole_target[ptr] + .path_list[.len]);
494 0816 5      whole_target[.len] = .path_list[.len] + .target_name[.len];
495 0817 5      dsc_cnt = 1;
496 0818 5      path_len = .whole_target[.len];
497 0819 5      path_ptr = .whole_target[ptr];
498 0820 6      loop (
499 0821 6      bind
500 0822 6      dsc_ptr = target_tsb[tsb_q_tkn_dsc, .dsc_cnt]: descriptor;
501 0823 7      if (.path_len neg 0) and (ch$rchar(.path_ptr) eq '_') then (
502 0824 7      whole_target[.len] = .whole_target[.len] - 1;
503 0825 7      ch$move((path_len = .path_len - 1), .path_ptr + 1, .path_ptr);
504 0826 6      );
505 0827 6      dsc_ptr[.len] = .path_len;
506 0828 6      dsc_ptr[ptr] = .path_ptr;
507 0829 6      exitif ((path_ptr = ch$find_ch(.path_len, .path_ptr, ':')) eq 0);
508 0830 6      path_ptr = .path_ptr + 2;
509 0831 6      dsc_ptr[.len] = .path_ptr - .dsc_ptr[ptr];
510 0832 6      path_len = .path_len - .dsc_ptr[.len];
511 0833 6      dsc_cnt = .dsc_cnt + 1;
512 0834 5      );
513 0835 5      target_tsb[tsb_w_tkn_count] = .dsc_cnt;
514 0836 4 );
515 0837 4
516 0838 3 end;
517 0839 3
518 0840 3 ! Now we have to do some more work if it's a link to a remote user.
519 0841 3
520 0842 4 if .target_tsb[tsb_v_user] then (
521 0843 4
522 0844 4      ! Now we send a special steering message to the network slave,
523 0845 4      ! asking it to verify the existence of the target user.
524 0846 4      ! It will send us back a status from that verification.
525 0847 4
526 0848 4      local
527 0849 4      local_described_buffer(verify_status, 4);
528 0850 4
529 0851 4      phn$send_smb(.tp, smb_slave_verify, target_tsb[tsb_q_tkn_dsc, 0]);
530 0852 4      status = phn$read_slave(.tp[pub_w_channel], verify_status, true);
531 0853 5      if .status nequ phn$ok then (
532 0854 5      phn$break_link(.tp, smb_slave_done);
533 0855 5      phn$inform(.status);
534 0856 5      return .status;
535 0857 4      );
536 0858 5      if ..verify_status[ptr] nequ phn$ok then (
```



```

: 537      0859 5      phn$break_link(.tp,smb_slave_done);
: 538      0860 5      phn$inform(..verify_status[ptr]);
: 539      0861 5      return ..verify_status[ptr];
: 540      0862 4      );
: 541      0863 4
: 542      0864 3 );
: 543      0865 3
: 544      0866 3 phn$inform(0);
: 545      0867 3 return phn$_ok;
: 546      0868 2 end;
: 547      0869 2
: 548      0870 1 end;

```

```

.PSECT $PLITS$,NOWRT,NOEXE,2
      000C 00044 P.AAG: .WORD 12
      0202 00046      .WORD 514
00000000' 00048      .ADDRESS USER_NAME+8
00000000' 0004C      .ADDRESS USER_NAME
      0004 00050      .WORD 4
      0303 00052      .WORD 771
00000000' 00054      .ADDRESS PARENT_PID
00000000 00058      .LONG 0
      0007 0005C      .WORD 7
      031D 0005E      .WORD 797
00000000' 00060      .ADDRESS TERM_NUMBER+8
00000000' 00064      .ADDRESS TERM_NUMBER
00000000 00068      .LONG 0
22 3D 39 32 22 0006C P.AAH: .ASCII '\29='\

```

```

.PSECT $OWNS$,NOEXE,2
0000000C 00109      .BLKB 3
0000000C 0010C USER_NAME:
      00000000' 00110      .LONG 12
      00114      .ADDRESS USER_NAME+8
      00114      .BLKB 12
00120 PARENT_PID:
      00000007 00124      .BLKB 4
      .LONG 7
00000000' 00128      .ADDRESS TERM_NUMBER+8
      0012C      .BLKB 7
      00133      .BLKB 1
0000 00134 PATH_LIST:
      02 0E 00136      .WORD 0
00000000 00138      .BYTE 14, 2
      .LONG 0

```

```

GET_PROC= P.AAG
.EXTRN SYSSGETJPIW, SYSSASSIGN
.EXTRN SYSSCREMBX, SYSSDELMBX
.EXTRN SYSTRNLNM

```

```

.PSECT $CODE$,NOWRT,2

```

				OFFC	00000	.ENTRY	PHN\$ESTAB_LINK, Save R2,R3,R4,R5,R6,R7,R8,-	
						MOVAB	R9,R10,R11	0567
						MOVQ	-292(SP), SP	
						CALLS	TARGET SPEC, -(SP)	0616
0000G						CALLS	#2, PHN\$MAKE_PUB	
						MOVL	R0, STATUS	
00000000C						CMPL	STATUS, #PHN\$_BADSPEC	0617
						BNEQ	1\$	
						PUSHL	#PHN\$_BADSPEC	0618
0000G						CALLS	#1, PHN\$INFORM	
						MOVL	#PHN\$_BADSPEC, R0	0619
						RET		
						BLBS	STATUS, 2\$	0621
						PUSHL	STATUS	
00000000G						CALLS	#1, LIB\$SIGNAL	
						MOVL	@PUB ADDRESS, TP	0622
						MOVAB	12(TP), R11	0628
						MOVZWL	2(R11), R0	0629
						MOVAB	4(R11)[R0], R8	
						BLBC	(R11), 3\$	0631
						BRW	17\$	
1A						BBS	#1, (R11), 4\$	0636
						PUSHL	TP	0637
0000G						CALLS	#1, PHN\$KILL_PUB	
						PUSHL	#PHN\$_NEEDUSER	0638
0000G						CALLS	#1, PHN\$INFORM	
						MOVL	#PHN\$_NEEDUSER, R0	0639
						RET		
						MOVL	PHN\$GQ_PUBHEAD, OP	0645
						PUSHAB	12(OP)	0646
						PUSHL	R11	
0000G						CALLS	#2, PHN\$CMP_TARGET	
						BLBC	R0, 6\$	
						MOVZWL	244(OP), R0	0652
						BNEQ	5\$	
						BRW	12\$	
00F4						MOVW	R0, 244(TP)	0653
						BRW	32\$	0655
						CLRQ	USABLE	0663
						MNEGL	#1, WILD_PID	0664
08						CLRQ	-(SP)	0672
						CLRL	-(SP)	
						PUSHAB	GET PROC	
						CLRL	-(SP)	
						PUSHAB	WILD_PID	
						PUSHL	#1	
00000000G						CALLS	#7, SYS\$GETJPIW	
						MOVL	R0, STATUS	
000009A8						CMPL	STATUS, #2472	0674
						BEQL	8\$	
						CMPL	STATUS, #1	0679
						BNEQ	7\$	
0000*	CF					CMPC5	(R8), @4(R8), #32, USER_NAME, @USER_NAME+4	0681
						BNEQ	7\$	
						TSTL	PARENT_PID	0682

			C6	12	000D2	BNEQ	7\$			
		0000'	CF	B5	000D4	TSTW	TERM_NUMBER			0683
			CO	13	000D8	BEQL	7\$			
			55	D6	000DA	INCL	POTENTIAL			0688
	7E	1000	8F	3C	000DC	MOVZWL	#4096, -(SP)			0689
		0000'	CF	9F	000E1	PUSHAB	TERM_NUMBER			
0000G	CF		02	FB	000E5	CALLS	#2, PHN\$TERM_CHARACTERISTIC			
	AD		50	E9	000EA	BLBC	R0, 7\$			
			54	D6	000ED	INCL	USABLE			0690
			A9	11	000EF	BRB	7\$			0664
			55	D5	000F1	8\$: TSTL	POTENTIAL			0698
			1A	12	000F3	BNEQ	9\$			
0000G	CF		5A	DD	000F5	PUSHL	TP			0699
		00000000G	01	FB	000F7	CALLS	#1, PHN\$KILL_PUB			
0000G	CF		8F	DD	000FC	PUSHL	#PHN\$NOPROC			0700
		00000000G	01	FB	00102	CALLS	#1, PHN\$INFORM			
	50		8F	DD	00107	MOVL	#PHN\$NOPROC, R0			0701
				04	0010E	RET				
			54	D5	0010F	9\$: TSTL	USABLE			0703
			1A	12	00111	BNEQ	10\$			
			5A	DD	00113	PUSHL	TP			0704
0000G	CF		01	FB	00115	CALLS	#1, PHN\$KILL_PUB			
		00000000G	8F	DD	0011A	PUSHL	#PHN\$TARGTERM			0705
0000G	CF		01	FB	00120	CALLS	#1, PHN\$INFORM			
	50		8F	DD	00125	MOVL	#PHN\$TARGTERM, R0			0706
				04	0012C	RET				
D4	AD		24	DD	0012D	10\$: MOVL	#36, MBX_NAME			0715
D8	AD	DC	AD	9E	00131	MOVAB	MBX_NAME+8, MBX_NAME+4			
		D4	AD	9F	00136	PUSHAB	MBX_NAME			0717
			58	DD	00139	PUSHL	R8			
0000V	CF		02	FB	0013B	CALLS	#2, PHN\$MBX_NAME			
			7E	7C	00140	CLRQ	-(SP)			0719
	52	00F4	CA	9E	00142	MOVAB	244(TP), R2			
			52	DD	00147	PUSHL	R2			
		D4	AD	9F	00149	PUSHAB	MBX_NAME			
00000000G	00		04	FB	0014C	CALLS	#4, SYS\$ASSIGN			
	6E		50	DD	00153	MOVL	R0, STATUS			
00000908	8F		6E	D1	00156	CMPL	STATUS, #2312			0720
			0E	13	0015D	BEQL	11\$			
00000144	8F		6E	D1	0015F	CMPL	STATUS, #324			0721
			05	13	00166	BEQL	11\$			
	57		6E	E9	00168	BLBC	STATUS, 15\$			0722
			5E	11	0016B	BRB	16\$			0723
		D4	AD	9F	0016D	11\$: PUSHAB	MBX_NAME			0732
			7E	7C	00170	CLRQ	-(SP)			
			7E	D4	00172	CLRL	-(SP)			
	7E	0100	8F	3C	00174	MOVZWL	#256, -(SP)			
			52	DD	00179	PUSHL	R2			
			01	DD	0017B	PUSHL	#1			
00000000G	00		07	FB	0017D	CALLS	#7, SYS\$CREMBX			
	6E		50	DD	00184	MOVL	R0, STATUS			
	24		6E	D1	00187	CMPL	STATUS, #36			0733
			1A	12	0018A	BNEQ	13\$			
			5A	DD	0018C	12\$: PUSHL	TP			0734
0000G	CF		01	FB	0018E	CALLS	#1, PHN\$KILL_PUB			
		00000000G	8F	DD	00193	PUSHL	#PHN\$NOPRIV			0735
0000G	CF		01	FB	00199	CALLS	#1, PHN\$INFORM			

				50	00000000G	8F	D0	0019E	MOVL	#PHN\$_NOPRIV, R0		0736
							04	001A5	RET			
				09			6E	E8 001A6	13\$:	BLBS	STATUS, 14\$	0738
							6E	DD 001A9		PUSHL	STATUS	
				00000000G	00		01	FB 001AB		CALLS	#1, LIB\$SIGNAL	
					7E		62	3C 001B2	14\$:	MOVZWL	(R2), -(SP)	0739
				00000000G	00		01	FB 001B5		CALLS	#1, SYS\$DELMBX	
					6E		50	D0 001BC		MOVL	R0, STATUS	
					09		6E	E8 001BF		BLBS	STATUS, 16\$	0740
							6E	DD 001C2	15\$:	PUSHL	STATUS	
				00000000G	00		01	FB 001C4		CALLS	#1, LIB\$SIGNAL	
							01	DA 31 001CB	16\$:	BRW	32\$	0741
					00000000G		8F	DD 001CE	17\$:	PUSHL	#PHN\$ DECNETLINK	0751
				0000G	CF		01	FB 001D4		CALLS	#1, PHN\$INFORM	
					56	04	AB	9E 001D9		MOVAB	4(R11), R6	0756
				OC	AE	FF	8F	9A 001DD		MOVZBL	#255, SPECIFIER_BUF	0760
				10	AE	14	AE	9E 001E2		MOVAB	SPECIFIER_BUF+8, SPECIFIER_BUF+4	
					50	F0	AD	9E 001E7		MOVAB	TRNLNMLST, \$\$ITMBLKPTR	0768
				80	000200FF		8F	D0 001EB		MOVL	#131327, (\$\$ITMBLKPTR)+	
				80		10	AE	D0 001F2		MOVL	SPECIFIER_BUF+4, (\$\$ITMBLKPTR)+	
				80		OC	AE	9E 001F6		MOVAB	SPECIFIER_BUF, (\$\$ITMBLKPTR)+	
							80	D4 001FA		CLRL	(\$\$ITMBLKPTR)+	
						F0	AD	9F 001FC		PUSHAB	TRNLNMLST	0773
							7E	D4 001FF		CLRL	-(SP)	
						0000'	CF	9F 00201		PUSHAB	SLAVE TASK DESC	
						0000'	CF	9F 00205		PUSHAB	LNLM PROCESS DESC	
				14	AE	02000000	8F	D0 00209		MOVL	#33554432, 20(SP)	
						14	AE	9F 00211		PUSHAB	20(SP)	
				00000000G	00		05	FB 00214		CALLS	#5, SYS\$TRNLNM	
					6E		50	D0 0021B		MOVL	R0, STATUS	
					01		6E	D1 0021E		CMPL	STATUS, #1	0775
							27	13 00221		BEQL	18\$	
				OC	AE	FF	8F	9B 00223		MOVZBW	#255, SPECIFIER_BUF	0777
						F0	AD	9F 00228		PUSHAB	TRNLNMLST	0781
							7E	D4 0022B		CLRL	-(SP)	
						0000'	CF	9F 0022D		PUSHAB	SLAVE TASK DESC	
						0000'	CF	9F 00231		PUSHAB	LNLM JOB DESC	
				14	AE	02000000	8F	D0 00235		MOVL	#33554432, 20(SP)	
						14	AE	9F 0023D		PUSHAB	20(SP)	
				00000000G	00		05	FB 00240		CALLS	#5, SYS\$TRNLNM	
					6E		50	D0 00247		MOVL	R0, STATUS	
					01		6E	D1 0024A	18\$:	CMPL	STATUS, #1	0784
							31	13 0024D		BEQL	20\$	
					59		66	3C 0024F		MOVZWL	(R6), R9	0785
					50		68	3C 00252		MOVZWL	(R8), R0	
					59		50	C2 00255		SUBL2	R0, R9	
				04	AE	FF	8F	9A 00258		MOVZBL	#255, 4(SP)	0787
					57	10	AE	D0 0025D		MOVL	SPECIFIER_BUF+4, R7	
04	AE		20	04	B6		59	2C 00261		MOVCS	R9, @4(R6), #32, 4(SP), (R7)	
							67	00268				
							10	18 00269		BGEQ	19\$	
					57		59	C0 0026B		ADDL2	R9, R7	
				04	AE		59	C2 0026E		SUBL2	R9, 4(SP)	
04	AE		20	0000'	CF		05	2C 00272		MOVCS	#5, P.AAH, #32, 4(SP), (R7)	
							67	0027A				
				OC	AE		05	A1 0027B	19\$:	ADDW3	#5, R9, SPECIFIER_BUF	0788
						0000'	CF	94 00280	20\$:	CLRB	PATH_ERROR_DONE	0795

				FD4D	CF	9F	00284	PUSHAB	PATH_ERROR		0796
					7E	D4	00288	CLRL	-(SPT)		
				0000'	CF	9F	0028A	PUSHAB	PATH_LIST		
					7E	D4	0028E	CLRL	-(SPT)		
				00F4	CA	9F	00290	PUSHAB	244(TP)		
				20	AE	9F	00294	PUSHAB	SPECIFIER_BUF		
		00000000G	00		06	FB	00297	CALLS	#6, UNS\$NET_CONNECT		
			6E		50	D0	0029E	MOVL	R0, STATUS		
			12		6E	E8	002A1	BLBS	STATUS, 22\$		0798
					5A	DD	002A4	PUSHL	TP		0799
		0000G	CF		01	FB	002A6	CALLS	#1, PHN\$KILL_PUB		
			03	0000'	CF	E9	002AB	BLBC	PATH_ERROR_DONE, 21\$		0800
					00CA	31	002B0	BRW	29\$		
					00C0	31	002B3	BRW	28\$		
			57	0000'	CF	3C	002B6	MOVZWL	PATH_LIST, R7		0807
			59		66	3C	002BB	MOVZWL	(R6), R9		0808
			50		68	3C	002BE	MOVZWL	(R8), R0		
			59		50	C2	002C1	SUBL2	R0, R9		
59		20	0000'	DF	57	2D	002C4	CMPC5	R7, @PATH_LIST+4, #32, R9, @4(R6)		0807
					04	B6	002CB				
					68	13	002CD	BEQL	27\$		
	10	BE	04	B8	68	28	002CF	MOV3	(R8), @4(R8), @SPECIFIER_BUF+4		0812
	04	B6	0000'	DF	57	28	002D5	MOV3	R7, @PATH_LIST+4, @4(R6)		0813
	04	B647	10	BE	68	28	002DC	MOV3	(R8), @SPECIFIER_BUF+4, @4(R6)[R7]		0815
		66		57	68	A1	002E3	ADDW3	(R8), R7, (R6)		0816
				59	01	D0	002E7	MOVL	#1, DSC_CNT		0817
			04	AE	66	3C	002EA	MOVZWL	(R6), PATH_LEN		0818
				58	04	A6	002EE	MOVL	4(R6), PATH_PTR		0819
				57	04	AB49	7E	002F2	MOVAQ	4(R11)[DSC_CNT], R7	0822
					04	AE	D5	002F7	TSTL	PATH_LEN	0823
					11	13	002FA	BEQL	24\$		
			5F	8F	68	91	002FC	CMPB	(PATH_PTR), #95		
					0B	12	00300	BNEQ	24\$		
					66	B7	00302	DECW	(R6)		0824
					04	AE	D7	00304	DECL	PATH_LEN	0825
			68	01	04	AE	28	00307	MOV3	PATH_LEN, 1(PATH_PTR), (PATH_PTR)	
				67	04	AE	B0	0030D	MOVW	PATH_LEN, (R7)	0827
				04	58	D0	00311	MOVL	PATH_PTR, 4(R7)		0828
			68	04	3A	3A	00315	LOCC	#58, PATH_LEN, (PATH_PTR)		0829
					02	12	0031A	BNEQ	25\$		
					51	D4	0031C	CLRL	R1		
				58	51	D0	0031E	MOVL	R1, PATH_PTR		
					13	13	00321	BEQL	26\$		
				58	02	C0	00323	ADDL2	#2, PATH_PTR		0830
			67	58	04	A7	A3	00326	SUBW3	4(R7), PATH_PTR, (R7)	0831
				50	67	3C	0032B	MOVZWL	(R7), R0		0832
				04	50	C2	0032E	SUBL2	R0, PATH_LEN		
					59	D6	00332	INCL	DSC_CNT		0833
					BC	11	00334	BRB	23\$		0819
				02	59	B0	00336	MOVW	DSC_CNT, 2(R11)		0835
			63	6B	01	E1	0033A	BBC	#1, (R11), 31\$		0842
				F4	04	D0	0033E	MOVL	#4, VERIFY_STATUS		0849
				F8	AD	9E	00342	MOVAB	VERIFY_STATUS+8, VERIFY_STATUS+4		
					56	DD	00347	PUSHL	R6		0851
					07	DD	00349	PUSHL	#7		
					5A	DD	0034B	PUSHL	TP		
			0000V	CF	03	FB	0034D	CALLS	#3, PHN\$SEND_SMB		

			01	DD	00352		PUSHL	#1		: 0852
		F4	AD	9F	00354		PUSHAB	VERIFY STATUS		:
		00F4	CA	3C	00357		MOVZWL	244(TPT, -(SP)		:
0000G	7E		03	FB	0035C		CALLS	#3, PHN\$READ_SLAVE		:
	CF		50	D0	00361		MOVL	R0, STATUS		:
00000000G	6E		6E	D1	00364		CMPL	STATUS, #PHN\$_OK		: 0853
	8F		14	13	0036B		BEQL	30\$		:
			0D	DD	0036D		PUSHL	#13		: 0854
			5A	DD	0036F		PUSHL	TP		:
0000V	CF		02	FB	00371		CALLS	#2, PHN\$BREAK_LINK		:
			6E	DD	00376	28\$:	PUSHL	STATUS		: 0855
0000G	CF		01	FB	00378		CALLS	#1, PHN\$INFORM		:
	50		6E	D0	0037D	29\$:	MOVL	STATUS, R0		: 0856
					04	00380	RET			:
00000000G	8F	F8	BD	D1	00381	30\$:	CMPL	@VERIFY_STATUS+4, #PHN\$_OK		: 0858
			16	13	00389		BEQL	31\$		:
			0D	DD	0038B		PUSHL	#13		: 0859
			5A	DD	0038D		PUSHL	TP		:
0000V	CF		02	FB	0038F		CALLS	#2, PHN\$BREAK_LINK		:
		F8	BD	DD	00394		PUSHL	@VERIFY_STATUS+4		: 0860
0000G	CF		01	FB	00397		CALLS	#1, PHN\$INFORM		:
	50	F8	BD	D0	0039C		MOVL	@VERIFY_STATUS+4, R0		: 0861
					04	003A0	RET			:
			7E	D4	003A1	31\$:	CLRL	-(SP)		: 0866
0000G	CF		01	FB	003A3		CALLS	#1, PHN\$INFORM		:
	50	00000000G	8F	D0	003A8	32\$:	MOVL	#PHN\$_OK, R0		: 0867
					04	003AF	RET			: 0870

; Routine Size: 944 bytes, Routine Base: \$CODE\$ + 0097

```
550 0871 1 %sbttl 'PHN$MBX_NAME - Build Mailbox Name'
551 0872 1 +-+
552 0873 1 Functional Description:
553 0874 1 This routine is called to build a mailbox name for use in creating
554 0875 1 a communication link mailbox. The name consists of a user
555 0876 1 name prefixed with 'PHN$'.
556 0877 1
557 0878 1 Formal Parameters:
558 0879 1 user_name Address of descriptor of user name.
559 0880 1 name_buf A descriptor for buffer to hold mailbox name.
560 0881 1 We set the length.
561 0882 1
562 0883 1 Implicit Inputs:
563 0884 1 global data
564 0885 1
565 0886 1 Implicit Outputs:
566 0887 1 global data
567 0888 1
568 0889 1 Returned Value:
569 0890 1 none
570 0891 1
571 0892 1 Side Effects:
572 0893 1 If a shared memory mailbox exists, a fatal error is signalled.
573 0894 1
574 0895 1 --
575 0896 1
576 0897 1
577 0898 2 global routine phn$mbx_name(user_name,name_buf): novalue = begin
578 0899 2
579 0900 2 bind
580 0901 2 user_name_dsc = .user_name: descriptor,
581 0902 2 name_buf_dsc = .name_buf: descriptor;
582 0903 2
583 0904 2 bind
584 0905 2 name_table = ch$transtable(
585 0906 2 rep 36 of ('_'),
586 0907 2 '$',
587 0908 2 rep 11 of (' '),
588 0909 2 '0','1','2','3','4','5','6','7','8','9',
589 0910 2 rep 7 of (' '),
590 0911 2 'A','B','C','D','E','F','G','H','I','J','K','L','M',
591 0912 2 'N','O','P','Q','R','S','T','U','V','W','X','Y','Z',
592 0913 2 rep 6 of (' '),
593 0914 2 'A','B','C','D','E','F','G','H','I','J','K','L','M',
594 0915 2 'N','O','P','Q','R','S','T','U','V','W','X','Y','Z',
595 0916 2 rep 5 of ('_'));
596 0917 2
597 0918 2 local
598 0919 2 status: long,
599 0920 2 local_described_buffer(shared_mailbox_name,4+4+16);
600 0921 2
601 0922 2
602 0923 2 ! To build the mailbox name, we concatenate the prefix and the user name.
603 0924 2 ! Then we run it through the translation table, converting any illegal
604 0925 2 ! characters to underscores.
605 0926 2
606 0927 2 ch$copy(4,uplit byte('PHN$'), .user_name_dsc[len],.user_name_dsc[ptr],
```

```

: 607      0928 2      ' ',.name_buf_dsc[len],.name_buf_dsc[ptr]);
: 608      0929 2      name_buf_dsc[len]= 4 + .user_name_dsc[len];
: 609      0930 2      ch$translate(name_table,.name_buf_dsc[len],.name_buf_dsc[ptr],' ',
: 610      0931 2      .name_buf_dsc[len],.name_buf_dsc[ptr]);
: 611      0932 2
: 612      0933 2      ! Now build the name of the corresponding shared memory mailbox,
: 613      0934 2      ! MBX$PHN$username. If a logical name exists by that name, then so
: 614      0935 2      ! does a mailbox. We can't allow that, so signal a fatal error.
: 615      0936 2
: 616      0937 2      begin
: 617      0938 2      local
: 618      0939 2      trlnmlst : $itmlst_decl(items=1),
: 619      0940 2      local_described_buffer(result_buffer,nam$c_maxrss);
: 620      0941 2
: 621      0942 2      ch$copy(4,uplit byte('MBX$'),.name_buf_dsc[len],.name_buf_dsc[ptr],
: 622      0943 2      .shared_mailbox_name[len],.shared_mailbox_name[ptr]);
: 623      0944 2      shared_mailbox_name[len] = 4 + .name_buf_dsc[len];
: 624      0945 2
: 625      P 0946 2      $itmlst_init(itmlst=trlnmlst,
: 626      P 0947 2      (itmcod=lnm$string,bufadr=.result_buffer[ptr],
: 627      0948 2      bufsiz=nam$c_maxrss,retlen=result_buffer));
: 628      0949 2
: 629      P 0950 2      status = $strlnm(attr=%ref(lnm$c_case blind),
: 630      P 0951 2      tabnam=$descriptor('LNM$SYSTEM'),
: 631      P 0952 2      lognam=shared_mailbox_name,
: 632      P 0953 2      acmode=%ref(psl$c_exec),
: 633      0954 2      itmlst=trlnmlst);
: 634      0955 2      if .status eqlu ss$normal then
: 635      0956 2      signal(phn$_sharedmbx)
: 636      0957 2      else
: 637      0958 2      if .status nequ ss$nolognam
: 638      0959 2      then check(.status);
: 639      0960 2      end;
: 640      0961 2
: 641      0962 2      return;
: 642      0963 2
: 643      0964 1      end;

```

```

.PSECT $SPLITS,NOWRT,NOEXE,2
00071 .BLKB 3
5F 00074 P.AAI: .ASCII \-
5F 00075 .ASCII \-
5F 00076 .ASCII \-
5F 00077 .ASCII \-
5F 00078 .ASCII \-
5F 00079 .ASCII \-
5F 0007A .ASCII \-
5F 0007B .ASCII \-
5F 0007C .ASCII \-
5F 0007D .ASCII \-
5F 0007E .ASCII \-
5F 0007F .ASCII \-
5F 00080 .ASCII \-
5F 00081 .ASCII \-

```



5F	00082	.ASCII	/-
5F	00083	.ASCII	/-
5F	00084	.ASCII	/-
5F	00085	.ASCII	/-
5F	00086	.ASCII	/-
5F	00087	.ASCII	/-
5F	00088	.ASCII	/-
5F	00089	.ASCII	/-
5F	0008A	.ASCII	/-
5F	0008B	.ASCII	/-
5F	0008C	.ASCII	/-
5F	0008D	.ASCII	/-
5F	0008E	.ASCII	/-
5F	0008F	.ASCII	/-
5F	00090	.ASCII	/-
5F	00091	.ASCII	/-
5F	00092	.ASCII	/-
5F	00093	.ASCII	/-
5F	00094	.ASCII	/-
5F	00095	.ASCII	/-
5F	00096	.ASCII	/-
5F	00097	.ASCII	/-
24	00098	.ASCII	/5
5F	00099	.ASCII	/-
5F	0009A	.ASCII	/-
5F	0009B	.ASCII	/-
5F	0009C	.ASCII	/-
5F	0009D	.ASCII	/-
5F	0009E	.ASCII	/-
5F	0009F	.ASCII	/-
5F	000A0	.ASCII	/-
5F	000A1	.ASCII	/-
5F	000A2	.ASCII	/-
5F	000A3	.ASCII	/-
30	000A4	.ASCII	/0
31	000A5	.ASCII	/1
32	000A6	.ASCII	/2
33	000A7	.ASCII	/3
34	000A8	.ASCII	/4
35	000A9	.ASCII	/5
36	000AA	.ASCII	/6
37	000AB	.ASCII	/7
38	000AC	.ASCII	/8
39	000AD	.ASCII	/9
5F	000AE	.ASCII	/-
5F	000AF	.ASCII	/-
5F	000B0	.ASCII	/-
5F	000B1	.ASCII	/-
5F	000B2	.ASCII	/-
5F	000B3	.ASCII	/-
5F	000B4	.ASCII	/-
41	000B5	.ASCII	/A
42	000B6	.ASCII	/B
43	000B7	.ASCII	/C
44	000B8	.ASCII	/D
45	000B9	.ASCII	/E
46	000BA	.ASCII	/F

.....

.....

47	000BB	.ASCII	\G\ .
48	000BC	.ASCII	\H\ .
49	000BD	.ASCII	\I\ .
4A	000BE	.ASCII	\J\ .
4B	000BF	.ASCII	\K\ .
4C	000C0	.ASCII	\L\ .
4D	000C1	.ASCII	\M\ .
4E	000C2	.ASCII	\N\ .
4F	000C3	.ASCII	\O\ .
50	000C4	.ASCII	\P\ .
51	000C5	.ASCII	\Q\ .
52	000C6	.ASCII	\R\ .
53	000C7	.ASCII	\S\ .
54	000C8	.ASCII	\T\ .
55	000C9	.ASCII	\U\ .
56	000CA	.ASCII	\V\ .
57	000CB	.ASCII	\W\ .
58	000CC	.ASCII	\X\ .
59	000CD	.ASCII	\Y\ .
5A	000CE	.ASCII	\Z\ .
5F	000CF	.ASCII	\_ .
5F	000D0	.ASCII	\_ .
5F	000D1	.ASCII	\_ .
5F	000D2	.ASCII	\_ .
5F	000D3	.ASCII	\_ .
5F	000D4	.ASCII	\_ .
41	000D5	.ASCII	\A\ .
42	000D6	.ASCII	\B\ .
43	000D7	.ASCII	\C\ .
44	000D8	.ASCII	\D\ .
45	000D9	.ASCII	\E\ .
46	000DA	.ASCII	\F\ .
47	000DB	.ASCII	\G\ .
48	000DC	.ASCII	\H\ .
49	000DD	.ASCII	\I\ .
4A	000DE	.ASCII	\J\ .
4B	000DF	.ASCII	\K\ .
4C	000E0	.ASCII	\L\ .
4D	000E1	.ASCII	\M\ .
4E	000E2	.ASCII	\N\ .
4F	000E3	.ASCII	\O\ .
50	000E4	.ASCII	\P\ .
51	000E5	.ASCII	\Q\ .
52	000E6	.ASCII	\R\ .
53	000E7	.ASCII	\S\ .
54	000E8	.ASCII	\T\ .
55	000E9	.ASCII	\U\ .
56	000EA	.ASCII	\V\ .
57	000EB	.ASCII	\W\ .
58	000EC	.ASCII	\X\ .
59	000ED	.ASCII	\Y\ .
5A	000EE	.ASCII	\Z\ .
5F	000EF	.ASCII	\_ .
5F	000F0	.ASCII	\_ .
5F	000F1	.ASCII	\_ .
5F	000F2	.ASCII	\_ .
5F	000F3	.ASCII	\_ .

.....

.....

				24	4E	48	50	000F4	P.AAJ:	.ASCII	\PHN\$\	:		
				24	58	42	4D	000F8	P.AAK:	.ASCII	\MBX\$\	:		
4D	45	54	53	59	53	24	4D	4E	4C	000FC	P.AAM:	.ASCII	\LNM\$SYSTEM\	:
										00106		.BLKB	2	:
								0000000A	00108	P.AAL:	.LONG	10	:	
								00000000	0010C		.ADDRESS	P.AAM	:	

NAME\_TABLE= P.AAI

.PSECT \$CODE\$,NOWRT,2

07FC 00000

.ENTRY PHN\$MBX\_NAME, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 0898  
R10

				5E	FECO	CE	9E	00002	MOVAB	-320(SP), SP	:		
				57	04	AC	D0	00007	MOVL	USER_NAME, R7	:	0901	
				56	08	AC	D0	0000B	MOVL	NAME_BUF, R6	:	0902	
			E0	AD		18	D0	0000F	MOVL	#24, SHARED_MAILBOX_NAME	:	0920	
			E4	AD	E8	AD	9E	00013	MOVAB	SHARED_MAILBOX_NAME+8, - SHARED_MAILBOX_NAME+4	:		
				5A		67	3C	00018	MOVZWL	(R7), R10	:	0927	
				59		66	3C	0001B	MOVZWL	(R6), R9	:	0928	
59		20		58	04	A6	D0	0001E	MOVL	4(R6), R8	:		
				CF		04	2C	00022	MOVCS	#4, P.AAJ, #32, R9, (R8)	:		
						68		00029			:		
						0D	18	0002A	BGEQ	1\$	:		
				58		04	C0	0002C	ADDL2	#4, R8	:		
				59		04	C2	0002F	SUBL2	#4, R9	:		
59		20	04	B7		5A	2C	00032	MOVCS	R10, @4(R7), #32, R9, (R8)	:		
						68		00038			:		
				66		04	A1	00039	1\$:	ADDW3	#4, (R7), (R6)	:	0929
0000'	CF	20	04	B6		66	2E	0003D	MOVTC	(R6), @4(R6), #32, NAME_TABLE, (R6), @4(R6)	:	0931	
				04		66		00045			:		
				08	AE	FF	8F	9A	00048	MOVZBL	#255, RESULT_BUFFER	:	0940
				0C	AE	10	AE	9E	0004D	MOVAB	RESULT_BUFFER+8, RESULT_BUFFER+4	:	
				59		66	3C	00052	MOVZWL	(R6), R9	:	0942	
				58	E0	AD	3C	00055	MOVZWL	SHARED_MAILBOX_NAME, R8	:	0943	
				57	E4	AD	D0	00059	MOVL	SHARED_MAILBOX_NAME+4, R7	:		
58		20	0000'	CF		04	2C	0005D	MOVCS	#4, P.AAK, #32, R8, (R7)	:		
						67		00064			:		
						0D	18	00065	BGEQ	2\$	:		
				57		04	C0	00067	ADDL2	#4, R7	:		
				58		04	C2	0006A	SUBL2	#4, R8	:		
58		20	04	B6		59	2C	0006D	MOVCS	R9, @4(R6), #32, R8, (R7)	:		
						67		00073			:		
			E0	AD		04	A1	00074	2\$:	ADDW3	#4, (R6), SHARED_MAILBOX_NAME	:	0944
				50	DO	AD	9E	00079	MOVAB	TRNLNMLST, \$\$ITMBLKPTR	:	0948	
				80	000200FF	8F	D0	0007D	MOVL	#131327, (\$\$ITMBLKPTR)+	:		
				80	0C	AE	D0	00084	MOVL	RESULT_BUFFER+4, (\$\$ITMBLKPTR)+	:		
				80	08	AE	9E	00088	MOVAB	RESULT_BUFFER, (\$\$ITMBLKPTR)+	:		
						80	D4	0008C	CLRL	(\$\$ITMBLKPTR)+	:		
					DO	AD	9F	0008E	PUSHAB	TRNLNMLST	:	0954	
				08	AE	01	D0	00091	MOVL	#1, 8(SP)	:		
					08	AE	9F	00095	PUSHAB	8(SP)	:		
					E0	AD	9F	00098	PUSHAB	SHARED_MAILBOX_NAME	:		
					0000'	CF	9F	0009B	PUSHAB	P.AAL	:		
				10	AE	02000000	8F	D0	0009F	MOVL	#33554432, 16(SP)	:	

LINKSUBS  
V04-000

LINKSUBS - Phone Link Subroutines  
PHN\$MBX\_NAME - Build Mailbox Name

E 5  
16-Sep-1984 02:11:44  
14-Sep-1984 12:53:27

VAX-11 Bliss-32 V4.0-742  
[PHONE.SRC]LINKSUBS.B32;1

Page 26  
(8)

00000000G	00	10	AE 9F 000A7	PUSHAB	16(SP)	:
	52		05 FB 000AA	CALLS	#5, SYS\$TRNLNM	:
	01		50 D0 000B1	MOVL	R0, STATUS	:
			52 D1 000B4	CMPL	STATUS, #1	0955
			08 12 000B7	BNEQ	3\$	:
		00000000G	8F DD 000B9	PUSHL	#PHN\$_SHAREDMBX	0956
			0E 11 000BF	BRB	4\$	:
000001BC	8F		52 D1 000C1 3\$:	CMPL	STATUS, #444	0958
			0C 13 000C8	BEQL	5\$	:
	09		52 E8 000CA	BLBS	STATUS, 5\$	0959
			52 DD 000CD	PUSHL	STATUS	:
00000000G	00		01 FB 000CF 4\$:	CALLS	#1, LIB\$SIGNAL	:
			04 000D6 5\$:	RET		0964

: Routine Size: 215 bytes, Routine Base: \$CODE\$ + 0447

```
645 0965 1 %sbttl 'PHN$BREAK_LINK - Break a Link'
646 0966 1 ++
647 0967 1 Functional Description:
648 0968 1 This routine is called to break a link between us and some other
649 0969 1 person or node.
650 0970 1
651 0971 1 Formal Parameters:
652 0972 1 target_pub The address of the PUB describing the link.
653 0973 1 smb_type The type code of the steering message to be sent
654 0974 1 as a reason for breaking the link.
655 0975 1 smb_msg An optional message text for the steering message.
656 0976 1
657 0977 1 Implicit Inputs:
658 0978 1 global data
659 0979 1
660 0980 1 Implicit Outputs:
661 0981 1 global data
662 0982 1
663 0983 1 Returned Value:
664 0984 1 none
665 0985 1
666 0986 1 Side Effects:
667 0987 1
668 0988 1 --
669 0989 1
670 0990 1
671 0991 2 global routine phn$break_link(target_pub,smb_type,smb_msg): novalue = begin
672 0992 2
673 0993 2 bind
674 0994 2 tp = .target_pub: pub,
675 0995 2 target_tsb = tp[pub_b_tsb]: tsb;
676 0996 2
677 0997 2 local
678 0998 2 status: long;
679 0999 2
680 1000 2 builtin
681 1001 2 argptr;
682 1002 2
683 1003 2
684 1004 2 ! First we send a message with the reason for breaking the link.
685 1005 2
686 1006 2 callg(argptr(),phn$send_smb);
687 1007 2
688 1008 2 ! If this is a remote link, we have to send the slave a special steering
689 1009 2 message to tell it to go away. If this message wasn't sent above, then
690 1010 2 do it now. Then we can clear away the network logical link.
691 1011 2
692 1012 3 if .target_tsb[tsb_v_remote] then (
693 1013 3 if .smb_type nequ smb_slave_done then
694 1014 3 phn$send_smb(tp,smb_slave_done);
695 1015 3 status = $dassign(chan=.tp[pub_w_channel]);
696 1016 3 check (.status);
697 1017 2 );
698 1018 2
699 1019 2 ! Finally we can kill the PUB representing the link we have broken.
700 1020 2
701 1021 2 phn$kill_pub(tp);
```

: 702  
: 703  
: 704  
1022 < return;  
1023 2  
1024 1 end;

					.EXTRN	SYSSDASSGN	
			0004	00000	.ENTRY	PHNSBREAK LINK, Save R2	: 0991
	52	04	AC	D0 00002	MOVL	TARGET PUB, R2	: 0994
0000V	CF		6C	FA 00006	CALLG	(AP), PHNSSEND_SMB	: 1006
	27	0C	A2	E9 0000B	BLBC	12(R2), 2\$	: 1012
	0D	08	AC	D1 0000F	CMPL	SMB_TYPE, #13	: 1013
			09	13 00013	BEQL	1\$	: 1014
			0D	DD 00015	PUSHL	#13	: 1014
			52	DD 00017	PUSHL	R2	: 1015
0000V	CF		02	FB 00019	CALLS	#2, PHNSSEND SMB	: 1015
	7E	00F4	C2	3C 0001E 1\$:	MOVZWL	244(R2), -(SP)	: 1016
00000000G	00		01	FB 00023	CALLS	#1, SYSSDASSGN	: 1021
	09		50	E8 0002A	BLBS	STATUS, 2\$	: 1021
			50	DD 0002D	PUSHL	STATUS	: 1021
00000000G	00		01	FB 0002F	CALLS	#1, LIBSSIGNAL	: 1021
			52	DD 00036 2\$:	PUSHL	R2	: 1021
0000G	CF		01	FB 00038	CALLS	#1, PHNSKILL_PUB	: 1021
			04	0003D	RET		: 1024

; Routine Size: 62 bytes, Routine Base: \$CODE\$ + 051E

```

: 706 1025 1 %sbttl 'PHN$BREAK_CALL - Break Link to Person We Are Calling'
: 707 1026 1 ++
: 708 1027 1 Functional Description:
: 709 1028 1 This routine is called to break the link to someone we are
: 710 1029 1 currently trying to call. It was made a seperate routine because
: 711 1030 1 it is done in a lot of places.
: 712 1031 1
: 713 1032 1 Formal Parameters:
: 714 1033 1 none
: 715 1034 1
: 716 1035 1 Implicit Inputs:
: 717 1036 1 global data
: 718 1037 1
: 719 1038 1 Implicit Outputs:
: 720 1039 1 global data
: 721 1040 1
: 722 1041 1 Returned Value:
: 723 1042 1 none
: 724 1043 1
: 725 1044 1 Side Effects:
: 726 1045 1
: 727 1046 1 --
: 728 1047 1
: 729 1048 1
: 730 1049 2 global routine phn$break_call: novalue = begin
: 731 1050 2
: 732 1051 2 local
: 733 1052 2 op: ref pub; ! Pointer to our PUB.
: 734 1053 2
: 735 1054 2
: 736 1055 2 ! We have to clear the calling flag and busylink in our PUB because we
: 737 1056 2 ! will no longer be calling the person. PHN$BREAK_LINK is then called to
: 738 1057 2 ! do its usual link-breaking stuff. Finally, we better cancel any outstanding
: 739 1058 2 ! timer requests, in case there is one waiting to ring the person's phone.
: 740 1059 2
: 741 1060 2 op = .phn$gq_pubhead[0];
: 742 1061 2 op[pub_v_calling] = false;
: 743 1062 2 phn$break_link(.op[pub_l_busylink],smb__hungup);
: 744 1063 2 op[pub_l_busylink] = 0;
: 745 1064 2 $cantim(T);
: 746 1065 2 return;
: 747 1066 2
: 748 1067 1 end;

```

```

                                .EXTRN  SYSSCANTIM
                                .ENTRY   PHN$BREAK CALL, Save R2
                                MOVL     PHN$GQ_PUBHEAD, OP
                                BICB2   #8, 240(OP)
                                PUSHL   #9
                                PUSHL   248(OP)
                                CALLS   #2, PHN$BREAK_LINK
                                CLRL    248(OP)
                                CLRQ    -(SP)
                                CALLS   #2, SYSSCANTIM
                                : 1049
                                : 1060
                                : 1061
                                : 1062
                                :
                                : 1063
                                : 1064
                                :

```

00F0	52	C2	0000G	CF	D0	00002	08	8A	00007
				09	DD	0000C	02	FB	00012
	AC	AF		00F8	C2	D4	00016	7E	7C
				00F8	C2	D4	00016	02	FB
00000000G	00			02	FB	0001C			

LINKSUBS  
V04-000

LINKSUBS - Phone Link Subroutines  
PHNSBREAK\_CALL - Break Link to Person We Are Ca

I 5  
16-Sep-1984 02:11:44  
14-Sep-1984 12:53:27

VAX-11 Bliss-32 V4.0-742  
[PHONE.SRC]LINKSUBS.B32;1

Page 30  
(10)

M  
V

04 00023

RET

: 1067

; Routine Size: 36 bytes,    Routine Base: \$CODE\$ + 055C



```
750 1068 1 %sbttl 'PHN$SEND_SMB - Send Steering Message'
751 1069 1 ++
752 1070 1 Functional Description:
753 1071 1 This routine is called to send a steering message to another person
754 1072 1 or node represented by a PUB. The message that is sent consists of
755 1073 1 up to three parts, as follows:
756 1074 1
757 1075 1 1. The 1-byte steering message code.
758 1076 1
759 1077 1 2. Our own node string, representing the sender of
760 1078 1 the message, followed by an eofrom character.
761 1079 1
762 1080 1 3. The optional message text.
763 1081 1
764 1082 1 Formal Parameters:
765 1083 1 target_pub Address of the PUB representing the destination.
766 1084 1 smb_type The steering message type code.
767 1085 1 smb_msg Address of descriptor of optional message text.
768 1086 1
769 1087 1 Implicit Inputs:
770 1088 1 global data
771 1089 1
772 1090 1 Implicit Outputs:
773 1091 1 global data
774 1092 1
775 1093 1 Returned Value:
776 1094 1 none
777 1095 1
778 1096 1 Side Effects:
779 1097 1
780 1098 1 --
781 1099 1
782 1100 1
783 1101 2 global routine phn$send_smb(target_pub,smb_type,smb_msg): novalue = begin
784 1102 2
785 1103 2 bind
786 1104 2 tp = .target_pub: pub,
787 1105 2 target_tsb = tp[pub_b_tsb]: tsb,
788 1106 2 smb_msg_dsc = .smb_msg: descriptor;
789 1107 2
790 1108 2 local
791 1109 2 status: long,
792 1110 2 op: ref pub, ! Pointer to our PUB.
793 1111 2 msg_buf: vector[phn$k_mbxsize,byte],
794 1112 2 buf_i: long,
795 1113 2 iosb: block[8,byte];
796 1114 2
797 1115 2 builtin
798 1116 2 nullparameter;
799 1117 2
800 1118 2
801 1119 2 ! First we have to build the message that we are going to send. Begin
802 1120 2 ! with the message type code.
803 1121 2
804 1122 2 buf_i = 0;
805 1123 2
806 1124 2 msg_buf[.buf_i] = .smb_type<0,8,0>;
```

```

807      1125 2 inc (buf_i);
808      1126 2
809      1127 2 ! Now include our spec, representing the sender of the message.
810      1128 2
811      1129 2 op = .phn$gq_pubhead[0];
812      1130 2 begin
813      1131 2 bind
814      1132 2     our_tsb = op[pub_b_tsb]: tsb,
815      1133 2     our_spec_dsc = our_tsb[tsb_q_tkndsc,0]: descriptor;
816      1134 2
817      1135 2 ! If the token count is above 2 we have a "pass through" connection so
818      1136 2 ! develop the reverse routing back to us. But, if we are a slave task then
819      1137 2 ! the routing has already been performed. In which case we don't do anything
820      1138 2 ! because we have arrived at the desired local location. Being the slave task
821      1139 2 ! is when the OP pub has the total routing contained. Any other time it would
822      1140 2 ! be our own home node specification.
823      1141 2
824      1142 3 if (.target_tsb[tsb_w_tkncount] gtru 2) and
825      1143 4     (.our_tsb[tsb_w_tkncount] lequ 2) then (
826      1144 4     decr_node_index from .target_tsb[tsb_w_tkncount] - 2 to 1 do
827      1145 5         begin
828      1146 5         bind
829      1147 5         node_dsc = target_tsb[tsb_q_tkndsc, .node_index]: descriptor;
830      1148 5         ch$move(.node_dsc[.len], .node_dsc[ptr], msg_buf[.buf_i]);
831      1149 5         buf_i = .buf_i + .node_dsc[.len];
832      1150 4     end;
833      1151 3 );
834      1152 2
835      1153 2 ch$move(.our_spec_dsc[.len], .our_spec_dsc[ptr], msg_buf[.buf_i]);
836      1154 2 buf_i = .buf_i + .our_spec_dsc[.len];
837      1155 2 msg_buf[.buf_i] = eofrom;
838      1156 2 inc (buf_i);
839      1157 2 end;
840      1158 2
841      1159 2 ! Finally, if there is optional message text, we have to move it into
842      1160 2 ! the buffer. Make sure we don't go off the end of the buffer.
843      1161 2
844      1162 3 if not nullparameter(3) then (
845      1163 3     local
846      1164 3         length: long;
847      1165 3
848      1166 3         length = minu(.smb_msg_dsc[.len], phn$k_mbxsize - .buf_i);
849      1167 3         ch$move(.length, .smb_msg_dsc[ptr], msg_buf[.buf_i]);
850      1168 3         buf_i = .buf_i + .length;
851      1169 2 );
852      1170 2
853      1171 2 ! Now we split up depending upon whether it's a local or remote message.
854      1172 2 ! If it's local, we can just send the message to the user's receive
855      1173 2 ! mailbox. If we get an error doing so, tell the user.
856      1174 2
857      1175 3 if not .target_tsb[tsb_v_remote] then (
858      1176 3     status = $qiow(chan=.tp[pub_w_channel],
859      1177 3         func=ios$writevblk + ios$m_now,
860      1178 3         iosb=iosb,
861      1179 3         p1=msg_buf,
862      1180 3         p2=.buf_i);
863      1181 3     if .status nequ ss$_normal or .iosb[0,0,16,0] nequ ss$_normal then

```

```

864      1182 3      phn$inform(phn$_linkerror);
865      1183 3      ) else (
866      1184 3
867      1185 3      ! It is a remote send. All we have to do is send the message over the
868      1186 3      ! logical link. If we get an error, tell the user.
869      1187 3
870      1188 3      status = $qiow(efn=phn$_dectefn,
871      1189 3      chan=.tp[pub_w_channel],
872      1190 3      func=ios$writevblk,
873      1191 3      iosb=iosb,
874      1192 3      p1=msg_buf,
875      1193 3      p2=.buf_i);
876      1194 3      if .status nequ ss$_normal or .iosb[0,0,16,0] nequ ss$_normal then
877      1195 3      phn$inform(phn$_linkerror);
878      1196 3      );
879      1197 2
880      1198 2      return;
881      1199 2
882      1200 1      end;

```

P  
P  
P  
P  
P

					.EXTRN	SYSSQIOW	
				OFFC 00000	.ENTRY	PHN\$SEND_SMB, Save R2,R3,R4,R5,R6,R7,R8,R9,-;	1101
	5E	FEF8	CE	9E 00002	MOVAB	-264(SP), SP	
	5B	04	AC	D0 00007	MOVL	TARGET PUB, R11	1104
	5A	0C	AB	9E 0000B	MOVAB	12(R11), R10	1105
		0C	AC	DD 0000F	PUSHL	SMB_MSG	1106
			56	D4 00012	CLRL	BUF_I	1122
	OC AE46	08	AC	90 00014	MOVB	SMB_TYPE, MSG_BUF[BUF_I]	1124
			56	D6 0001A	INCL	BUF_I	1125
	50	0000G	CF	D0 0001C	MOVL	PHN\$GQ PUBHEAD, OP	1129
	50		0C	C0 00021	ADDL2	#12, R0	1132
	59	04	A0	9E 00024	MOVAB	4(R0), R9	1133
	02	02	AA	B1 00028	CMPW	2(R10), #2	1142
			23	1B 0002C	BLEQU	3\$	
	02	02	A0	B1 0002F	CMPW	2(R0), #2	1143
			1D	1A 00032	BGTRU	3\$	
	58	02	AA	3C 00034	MOVZWL	2(R10), NODE_INDEX	1144
			58	D7 00038	DECL	NODE_INDEX	
			12	11 0003A	BRB	2\$	
	OC AE46	04	AA48	7E 0003C 1\$:	MOVAQ	4(R10)[NODE_INDEX], R7	1147
		04	67	28 00041	MOVCS	(R7), @4(R7), MSG_BUF[BUF_I]	1148
			67	3C 00048	MOVZWL	(R7), R0	1149
			50	C0 0004B	ADDL2	R0, BUF_I	
			58	F5 0004E 2\$:	SOBGTR	NODE_INDEX, 1\$	1144
	OC AE46	04	69	28 00051 3\$:	MOVCS	(R9), @4(R9), MSG_BUF[BUF_I]	1153
			69	3C 00058	MOVZWL	(R9), R0	1154
			50	C0 0005B	ADDL2	R0, BUF_I	
		OC AE46	94	0005E	CLRB	MSG_BUF[BUF_I]	1155
			56	D6 00062	INCL	BUF_I	1156
		03	6C	91 00064	CMPB	(APT), #3	1162
			29	1F 00067	BLSSU	5\$	
		OC	AC	D5 00069	TSTL	12(AP)	
			24	13 0006C	BEQL	5\$	

51	00000100	8F		56	C3	0006E	SUBL3	BUF_I, #256, R1	:	1166
		50	00	BE	3C	00076	MOVZWL	@0(SP), R0	:	
		51		50	D1	0007A	CMPL	R0, R1	:	
				03	1B	0007D	BLEQU	7\$	:	
		50		51	D0	0007F	MOVL	R1, R0	:	
		58		50	D0	00082	4\$:	MOV	R0, LENGTH	
57		6E		04	C1	00085	ADDL3	#4, (SP), R7	:	1167
OC AE46		97		58	28	00089	MOVCL3	LENGTH, @ (R7)+, MSG_BUF[BUF_I]	:	
		56		58	C0	0008F	ADDL2	LENGTH, BUF_I	:	1168
		27		6A	E8	00092	5\$:	BLBS	(R10), 6\$	1175
				7E	7C	00095	CLRQ	-(SP)	:	1180
				7E	7C	00097	CLRQ	-(SP)	:	
			20	56	DD	00099	PUSHL	BUF_I	:	
				AE	9F	0009B	PUSHAB	MSG_BUF	:	
			24	7E	7C	0009E	CLRQ	-(SP)	:	
		7E		AE	9F	000A0	PUSHAB	IOSB	:	
		7E	00F4	8F	9A	000A3	MOVZBL	#112, -(SP)	:	
				CB	3C	000A7	MOVZWL	244(R11), -(SP)	:	
				7E	D4	000AC	CLRL	-(SP)	:	
00000000G		00		0C	FB	000AE	CALLS	#12, SYSSQIOW	:	
		01		50	D1	000B5	CMPL	STATUS, #1	:	1181
				25	13	000B8	BEQL	7\$	:	
				29	11	000BA	BRB	8\$	:	1182
				7E	7C	000BC	6\$:	CLRQ	-(SP)	1193
				7E	7C	000BE	CLRQ	-(SP)	:	
			20	56	DD	000C0	PUSHL	BUF_I	:	
				AE	9F	000C2	PUSHAB	MSG_BUF	:	
			24	7E	7C	000C5	CLRQ	-(SP)	:	
				AE	9F	000C7	PUSHAB	IOSB	:	
				30	DD	000CA	PUSHL	#48	:	
		7E	00F4	CB	3C	000CC	MOVZWL	244(R11), -(SP)	:	
				04	DD	000D1	PUSHL	#4	:	
00000000G		00		0C	FB	000D3	CALLS	#12, SYSSQIOW	:	
		01		50	D1	000DA	CMPL	STATUS, #1	:	1194
				06	12	000DD	BNEQ	8\$	:	
		01	04	AE	B1	000DF	7\$:	CMPW	IOSB, #1	
				0B	13	000E3	BEQL	9\$	:	
			00000000G	8F	DD	000E5	8\$:	PUSHL	#PHN\$ LINKERROR	1195
0000G	CF			01	FB	000EB	CALLS	#1, PHN\$INFORM	:	
				04	000F0	9\$:	RET		:	1200

; Routine Size: 241 bytes, Routine Base: \$CODE\$ + 0580

```

884 1201 1 %sbttl 'PHNS$FORCE_LINKS - Force Links to New Person'
885 1202 1 ++
886 1203 1 Functional Description:
887 1204 1 This routine is called when a new person enters the conversation
888 1205 1 as a result of answering our call. We need to inform everyone
889 1206 1 else in the conversation about the new person, and also inform
890 1207 1 the new person about them. This routine is the principle vehicle
891 1208 1 for setting up conference calls.
892 1209 1
893 1210 1 Formal Parameters:
894 1211 1 new_pub Address of new person's PUB.
895 1212 1
896 1213 1 Implicit Inputs:
897 1214 1 global data
898 1215 1
899 1216 1 Implicit Outputs:
900 1217 1 global data
901 1218 1
902 1219 1 Returned Value:
903 1220 1 none
904 1221 1
905 1222 1 Side Effects:
906 1223 1
907 1224 1 --
908 1225 1
909 1226 1
910 1227 2 global routine phn$force_links(new_pub): novalue = begin
911 1228 2
912 1229 2 bind
913 1230 2 np = .new_pub: pub,
914 1231 2 new_tsb = np[pub_b_tsb]: tsb;
915 1232 2
916 1233 2 local
917 1234 2 p: ref pub;
918 1235 2
919 1236 2
920 1237 2 ! We scan the PUB chain (but not our own), looking for people in the
921 1238 2 ! current conversation. Make sure not to find the new person's PUB.
922 1239 2
923 1240 2 p = .phn$gq_pubhead[0];
924 1241 3 until .p eq la phn$gq_pubhead do (
925 1242 3 if not .p[pub_v_temporary] and
926 1243 3 not .p[pub_v_uhaveheld] and
927 1244 4 (.p neqa np) then (
928 1245 4
929 1246 4 ! We found a PUB. Tell this person about the new guy.
930 1247 4 ! Tell the new guy about this person.
931 1248 4
932 1249 4 bind
933 1250 4 person_tsb = p[pub_b_tsb]: tsb;
934 1251 4
935 1252 4 phn$send_smb(.p,smb__forced_link,new_tsb [tsb_q_tkndsc,0]);
936 1253 4 phn$send_smb(np,smb__forced_link,person_tsb[tsb_q_tkndsc,0]);
937 1254 3 );
938 1255 3
939 1256 3 p = .p[pub_l_flink];
940 1257 2 );

```

: 941  
: 942  
: 943  
: 944  
1258 2  
1259 2 return;  
1260 2  
1261 1 end;

53	04	AC		000C 00000	.ENTRY PHNS\$FORCE_LINKS, Save R2,R3	: 1227
		52	0000G	0C C1 00002	ADDL3 #12, NEW_PUB, R3	: 1231
		50	0000G	DF D0 00007	MOVL @PHNS\$GQ_PUBHEAD, P	: 1240
		50		CF 9E 0000C 1\$:	MOVAB PHNS\$GQ_PUBHEAD, R0	: 1241
				52 D1 00011	CMPL P, R0	:
				2F 13 00014	BEQL 3\$	:
24	00F0	C2		02 E0 00016	BBS #2, 240(P), 2\$	: 1242
		1F	00F0	C2 EB 0001C	BLBS 240(P), 2\$	: 1243
	04	AC		52 D1 00021	CMPL P, NEW_PUB	: 1244
				19 13 00025	BEQL 2\$	:
			04	A3 9F 00027	PUSHAB 4(R3)	: 1252
				11 DD 0002A	PUSHL #17	:
				52 DD 0002C	PUSHL P	:
	FEDC	CF		03 FB 0002E	CALLS #3, PHNS\$SEND_SMB	:
			10	A2 9F 00033	PUSHAB 16(P)	: 1253
				11 DD 00036	PUSHL #17	:
			04	AC DD 00038	PUSHL NEW_PUB	:
	FECF	CF		03 FB 0003B	CALLS #3, PHNS\$SEND_SMB	:
		52		62 D0 00040 2\$:	MOVL (P), P	: 1256
				C7 11 00043	BRB 1\$	: 1241
				04 00045 3\$:	RET	: 1261

; Routine Size: 70 bytes, Routine Base: \$CODE\$ + 0671

```
946 1262 1 %sbttl 'PHN$FORCED_LINK - Handle Forced Link Message'
947 1263 1 ++
948 1264 1 Functional Description:
949 1265 1 This steering message routine handles the forced_link message,
950 1266 1 which someone sends us when they want to force us to establish
951 1267 1 a link to a third party. This is done by the person responsible
952 1268 1 for setting up a conference call.
953 1269 1
954 1270 1 Formal Parameters:
955 1271 1 from_msg The address of a descriptor of the message. It
956 1272 1 consists, as usual, of the sender's node/user name
957 1273 1 spec followed by an eofrom character. Following this
958 1274 1 is the node/user name spec of the person we are being
959 1275 1 forced to link to.
960 1276 1
961 1277 1 Implicit Inputs:
962 1278 1 global data
963 1279 1
964 1280 1 Implicit Outputs:
965 1281 1 global data
966 1282 1
967 1283 1 Returned Value:
968 1284 1 none
969 1285 1
970 1286 1 Side Effects:
971 1287 1
972 1288 1 --
973 1289 1
974 1290 1
975 1291 2 global routine phn$forced_link(from_msg): novalue = begin
976 1292 2
977 1293 2 bind
978 1294 2 from_msg_dsc = .from_msg: descriptor;
979 1295 2
980 1296 2 local
981 1297 2 status: long,
982 1298 2 third_party_dsc: descriptor,
983 1299 2 sender_tsb: tsb,
984 1300 2 third_party_tsb: tsb,
985 1301 2 tp: ref pub, ! Pointer to third party's PUB.
986 1302 2 p: ref pub;
987 1303 2
988 1304 2
989 1305 2 ! We begin by rebuilding the from_msg descriptor so that it only describes
990 1306 2 ! the sender's spec. We build a new descriptor, third_party_dsc, to describe
991 1307 2 ! the spec of the third party.
992 1308 2
993 1309 2 third_party_dsc[ptr] = ch$find_ch(.from_msg_dsc[len],.from_msg_dsc[ptr],
994 1310 2 eofrom) + 1;
995 1311 3 third_party_dsc[len] = .from_msg_dsc[len] - (.third_party_dsc[ptr] -
996 1312 2 .from_msg_dsc[ptr]);
997 1313 2 from_msg_dsc[len] = .from_msg_dsc[len] - .third_party_dsc[len] - 1;
998 1314 2
999 1315 2 ! Now we make TSBs for both the sender and the third party, because we need
1000 1316 2 ! to parse both of their specs.
1001 1317 2
1002 1318 2 status = phn$make_tsb(from_msg_dsc, sender_tsb);
```

```

: 1003      1319 2 check (.status);
: 1004      1320 2 status = phn$make_tsb(third_party_dsc,third_party_tsb);
: 1005      1321 2 check (.status);
: 1006      1322 2
: 1007      1323 2 ! Now we have to scan the PUB chain and make sure that we do not already
: 1008      1324 2 ! have a link to the third party. If so, just ignore the message.
: 1009      1325 2
: 1010      1326 2 p = .phn$gq_pubhead[0];
: 1011      1327 2 until .p eq[a phn$gq_pubhead do (
: 1012      1328 3     if phn$cmp_target(p[pub_b_tsb],third_party_tsb) then
: 1013      1329 3         return;
: 1014      1330 3     p = .p[pub_l_flink];
: 1015      1331 2 );
: 1016      1332 2
: 1017      1333 2 ! Now we establish a link to the third party. We have to remember to
: 1018      1334 2 ! make their PUB permanent, and to assign them a viewport. If anything
: 1019      1335 2 ! prevents this, just bag the link.
: 1020      1336 2
: 1021      1337 2 status = phn$estab_link(third_party_tsb[tsb_q_tkndsc,0],tp);
: 1022      1338 2 if .status nequ phn$ok then
: 1023      1339 2     return;
: 1024      1340 2 tp[pub_v_temporary] = false;
: 1025      1341 2 status = phn$fresh_screen(false);
: 1026      1342 2 if .status nequ phn$ok then (
: 1027      1343 3     phn$break_link(.tp,smb__hungup);
: 1028      1344 3     return;
: 1029      1345 2 );
: 1030      1346 2
: 1031      1347 2 ! Finally, inform the user about the third party, including the sender's
: 1032      1348 2 ! name and the third party's name.
: 1033      1349 2
: 1034      1350 2 phn$inform(phn$_confcall,
: 1035      1351 2     sender_tsb[tsb_q_tkndsc,.sender_tsb [tsb_w_tkncount]],
: 1036      1352 2     third_party_tsb[tsb_q_tkndsc,.third_party_tsb[tsb_w_tkncount]]);
: 1037      1353 2 return;
: 1038      1354 2
: 1039      1355 1 end;

```

					003C 00000	.ENTRY PHN\$FORCED LINK, Save R2,R3,R4,R5	: 1291
		55	00000000G	8F	D0 00002	MOVL #PHN\$OK, R5	:
		54	00000000G	00	9E 00009	MOVAB LIB\$SIGNAL, R4	:
		5E	FE2C	CE	9E 00010	MOVAB -468(SP), SP	:
04	B2	52	04	AC	D0 00015	MOVL FROM MSG, R2	: 1294
		62		00	3A 00019	LOCC #0, (R2), @4(R2)	: 1309
				02	12 0001E	BNEQ 1\$	:
				51	D4 00020	CLRL R1	:
		FC	AD	01	A1 9E 00022	MOVAB 1(R1), THIRD_PARTY_DSC+4	: 1310
F8	50	04	A2	FC	AD C3 00027	SUBL3 THIRD_PARTY_DSC+4,-4(R2), R0	: 1312
	AD		50	62	A1 0002D	ADDW3 (R2), R0, THIRD_PARTY_DSC	: 1311
			50	62	3C 00032	MOVZWL (R2), R0	: 1313
			51	F8	AD 3C 00035	MOVZWL THIRD_PARTY_DSC, R1	:
			50	51	C2 00039	SUBL2 R1, R0	:
	62		50	01	A3 0003C	SUBW3 #1, R0, (R2)	:



		00E8	CE 9F 00040		PUSHAB	SENDER_TSB	:	1318	:
			52 DD 00044		PUSHL	R2	:		:
0000G	CF		02 FB 00046		CALLS	#2, PHN\$MAKE_TSB	:		:
	53		50 D0 0004B		MOVL	R0, STATUS	:		:
	05		53 E8 0004E		BLBS	STATUS, 2\$	:	1319	:
			53 DD 00051		PUSHL	STATUS	:		:
	64		01 FB 00053		CALLS	#1, LIB\$SIGNAL	:		:
		04	AE 9F 00056	2\$:	PUSHAB	THIRD_PARTY_TSB	:	1320	:
		F8	AD 9F 00059		PUSHAB	THIRD_PARTY_DSC	:		:
0000G	CF		02 FB 0005C		CALLS	#2, PHN\$MAKE_TSB	:		:
	53		50 D0 00061		MOVL	R0, STATUS	:		:
	05		53 E8 00064		BLBS	STATUS, 3\$	:	1321	:
			53 DD 00067		PUSHL	STATUS	:		:
	64		01 FB 00069		CALLS	#1, LIB\$SIGNAL	:		:
	52	0000G	CF D0 0006C	3\$:	MOVL	PHN\$GQ_PUBHEAD, P	:	1326	:
	50	0000G	CF 9E 00071	4\$:	MOVAB	PHN\$GQ_PUBHEAD, R0	:	1327	:
	50		52 D1 00076		CMPL	P, R0	:		:
			13 13 00079		BEQL	5\$	:		:
		04	AE 9F 0007B		PUSHAB	THIRD_PARTY_TSB	:	1328	:
		0C	A2 9F 0007E		PUSHAB	12(P)	:		:
0000G	CF		02 FB 00081		CALLS	#2, PHN\$CMP_TARGET	:		:
	55		50 E8 00086		BLBS	R0, 7\$	:		:
	52		62 D0 00089		MOVL	(P), P	:	1330	:
			E3 11 0008C		BRB	4\$	:	1327	:
			5E DD 0008E	5\$:	PUSHL	SP	:	1337	:
		0C	AE 9F 00090		PUSHAB	THIRD PARTY_TSB+4	:		:
F948	CF		02 FB 00093		CALLS	#2, PHN\$ESTAB_LINK	:		:
	53		50 D0 00098		MOVL	R0, STATUS	:		:
	55		53 D1 0009B		CMPL	STATUS, R5	:	1338	:
			3E 12 0009E		BNEQ	7\$	:		:
	52		6E D0 000A0		MOVL	TP, R2	:	1340	:
00F0	C2		04 8A 000A3		BICB2	#4, 240(R2)	:		:
			7E D4 000A8		CLRL	-(SP)	:	1341	:
0000G	CF		01 FB 000AA		CALLS	#1, PHN\$FRESH_SCREEN	:		:
	53		50 D0 000AF		MOVL	R0, STATUS	:		:
	55		53 D1 000B2		CMPL	STATUS, R5	:	1342	:
			0A 13 000B5		BEQL	6\$	:		:
			09 DD 000B7		PUSHL	#9	:	1343	:
			52 DD 000B9		PUSHL	R2	:		:
FDA7	CF		02 FB 000BB		CALLS	#2, PHN\$BREAK_LINK	:		:
			04 000C0		RET		:	1342	:
	50	06	AE 3C 000C1	6\$:	MOVZWL	THIRD_PARTY_TSB+2, R0	:	1352	:
		08	AE40 7F 000C5		PUSHAQ	THIRD_PARTY_TSB+4[R0]	:		:
	50	FF16	CD 3C 000C9		MOVZWL	SENDER_TSB+2, R0	:	1351	:
		FF18	CD40 7F 000CE		PUSHAQ	SENDER_TSB+4[R0]	:		:
		00000000G	8F DD 000D3		PUSHL	#PHN\$ CONFCALL	:	1352	:
0000G	CF		03 FB 000D9		CALLS	#3, PHN\$INFORM	:		:
			04 000DE	7\$:	RET		:	1355	:

; Routine Size: 223 bytes, Routine Base: \$CODE\$ + 06B7

; 1040 1356 1  
; 1041 1357 0 end eludom

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$PLITS	272	NOVEC,NOWRT, RD,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$OWNS	316	NOVEC, WRT, RD,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$CODES	1942	NOVEC,NOWRT, RD, EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	Symbols		Pages Mapped	Processing Time
	Total	Loaded Percent		
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	47 0	581	00:00.7

COMMAND QUALIFIERS

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

; Size: 1942 code + 588 data bytes  
 ; Run Time: 00:26.7  
 ; Elapsed Time: 01:36.1  
 ; Lines/CPU Min: 3044  
 ; Lexemes/CPU-Min: 32221  
 ; Memory Used: 298 pages  
 ; Compilation Complete

[Thumbnail 1]	[Thumbnail 2]	[Thumbnail 3]	[Thumbnail 4]	[Thumbnail 5]	[Thumbnail 6]	[Thumbnail 7]	[Thumbnail 8]	[Thumbnail 9]	[Thumbnail 10]	[Thumbnail 11]	[Thumbnail 12]	[Thumbnail 13]	[Thumbnail 14]	[Thumbnail 15]	[Thumbnail 16]	[Thumbnail 17]	[Thumbnail 18]	[Thumbnail 19]	[Thumbnail 20]	[Thumbnail 21]	[Thumbnail 22]	[Thumbnail 23]	[Thumbnail 24]	[Thumbnail 25]	[Thumbnail 26]	[Thumbnail 27]	[Thumbnail 28]	[Thumbnail 29]	[Thumbnail 30]	[Thumbnail 31]	[Thumbnail 32]	[Thumbnail 33]	[Thumbnail 34]	[Thumbnail 35]	[Thumbnail 36]	[Thumbnail 37]	[Thumbnail 38]	[Thumbnail 39]	[Thumbnail 40]	[Thumbnail 41]	[Thumbnail 42]	[Thumbnail 43]	[Thumbnail 44]	[Thumbnail 45]	[Thumbnail 46]	[Thumbnail 47]	[Thumbnail 48]	[Thumbnail 49]	[Thumbnail 50]	[Thumbnail 51]	[Thumbnail 52]	[Thumbnail 53]	[Thumbnail 54]	[Thumbnail 55]	[Thumbnail 56]	[Thumbnail 57]	[Thumbnail 58]	[Thumbnail 59]	[Thumbnail 60]	[Thumbnail 61]	[Thumbnail 62]	[Thumbnail 63]	[Thumbnail 64]	[Thumbnail 65]	[Thumbnail 66]	[Thumbnail 67]	[Thumbnail 68]	[Thumbnail 69]	[Thumbnail 70]	[Thumbnail 71]	[Thumbnail 72]	[Thumbnail 73]	[Thumbnail 74]	[Thumbnail 75]	[Thumbnail 76]	[Thumbnail 77]	[Thumbnail 78]	[Thumbnail 79]	[Thumbnail 80]	[Thumbnail 81]	[Thumbnail 82]	[Thumbnail 83]	[Thumbnail 84]	[Thumbnail 85]	[Thumbnail 86]	[Thumbnail 87]	[Thumbnail 88]	[Thumbnail 89]	[Thumbnail 90]	[Thumbnail 91]	[Thumbnail 92]	[Thumbnail 93]	[Thumbnail 94]	[Thumbnail 95]	[Thumbnail 96]	[Thumbnail 97]	[Thumbnail 98]	[Thumbnail 99]	[Thumbnail 100]
---------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	-----------------

PHONE LIS

NETSLAVE LIS

LINKSUBS LIS

PHONEMSGS LIS

STACKMDS LIS

TERMINAL LIS

MISCCNDS LIS

PUBSUBS LIS

INPUT LIS