


```

PPPPPPPP      HH      HH      000000      NN      NN      EEEEEEEEEEE
PPPPPPPP      HH      HH      000000      NN      NN      EEEEEEEEEEE
PP           PP  HH      HH      00      00      NN      NN      EE
PP           PP  HH      HH      00      00      NN      NN      EE
PP           PP  HH      HH      00      00      NNNN     NN      EE
PP           PP  HH      HH      00      00      NNNN     NN      EE
PPPPPPPP      HHHHHHHHHH  00      00      NN      NN      EEEEEEEEEEE
PPPPPPPP      HHHHHHHHHH  00      00      NN      NN      EEEEEEEEEEE
PP           HH      HH      00      00      NN      NNNN     EE
PP           HH      HH      00      00      NN      NNNN     EE
PP           HH      HH      00      00      NN      NN      EE
PP           HH      HH      00      00      NN      NN      EE
PP           HH      HH      000000      NN      NN      EEEEEEEEEEE
PP           HH      HH      000000      NN      NN      EEEEEEEEEEE

```

```

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

```



```

1 0001 0 %title 'PHONE - VAX/VMS Telephone Facility'
2 0002 0      module phone ( main=phn$phone,
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, Main Module
32 0032 1
33 0033 1 Abstract:      This facility allows the user to talk to other users on the
34 0034 1                local node or any other node. The goal of this facility
35 0035 1                is to simulate a real telephone as closely as possible,
36 0036 1                hopefully resulting in good human engineering.
37 0037 1
38 0038 1                Special thanks to Tim Halvorsen, author of TALK, the original
39 0039 1                VAX/VMS telephone facility.
40 0040 1
41 0041 1
42 0042 1 Environment:  Native, User mode. The following privileges are required for
43 0043 1                full operation; PHONE should be installed with them:
44 0044 1
45 0045 1                NETMBX  To call over the network.
46 0046 1                OPER    To ring a phone via broadcasting.
47 0047 1                PRMMBX  To talk to any other user.
48 0048 1                WORLD   To obtain information about other processes.
49 0049 1
50 0050 1                For remote communication, we assume all participating nodes
51 0051 1                have DECnet Phase III, with routing.
52 0052 1
53 0053 1 Author: Paul C. Anagnostopoulos, Creation Date: 18 November 1980
54 0054 1
55 0055 1 Modified By:
56 0056 1
57 0057 1                V03-004 BLS0251                Benn Schreiber                8-Dec-1983

```

PHONE
V04-000

PHONE - VAX/VMS Telephone Facility

D 11
16-Sep-1984 02:15:58
14-Sep-1984 12:53:28

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

P
V

:	58	0058	1	:	
:	59	0059	1	:	Increase size of mailbox name buffer to account for
:	60	0060	1	:	extra overhead characters added. Convert \$TRNLOG to
:	61	0061	1	:	\$TRNLNM.
:	62	0062	1	:	V03-003 PCA1004 Paul C. Anagnostopoulos 8-Nov-1982
:	63	0063	1	:	\$ASSIGN no longer returns SS\$_IVDEVNAM when assigning to
:	64	0064	1	:	a mailbox that doesn't exist. It now returns SS\$_NOSUCHDEV.
:	65	0065	1	:	
:	66	0066	1	:	V03-002 PCA1000 Paul C. Anagnostopoulos 5-Oct-1982
:	67	0067	1	:	Remove all references to CLISEND_PARSE, which is now obsolete.
:	68	0068	1	:	
:	69	0069	1	:	V03-001 PCA0045 Paul Anagnostopoulos 26-Mar-1982
:	70	0070	1	:	Major changes to convert from process name to user name.
:	71	0071	1	!--	

```

: 73 0072 1 %sbttl 'Module Declarations'
: 74 0073 1
: 75 0074 1 | Libraries and Requires:
: 76 0075 1 |
: 77 0076 1 |
: 78 0077 1 library 'sys$library:starlet.l32';
: 79 0078 1 literal global_data = 1;           ! To suppress generation of external
: 80 0079 1 require 'phone_req';             ! data declarations
: 81 0408 1
: 82 0409 1 |
: 83 0410 1 | Table of Contents:
: 84 0411 1 |
: 85 0412 1 |
: 86 0413 1 forward routine
: 87 0414 1     phn$phone: novalue,
: 88 0415 1     phn$init_main: novalue,
: 89 0416 1     phn$dcl_command_line: novalue,
: 90 0417 1     phn$queue_smb: novalue,
: 91 0418 1     phn$kill_smb: novalue,
: 92 0419 1     phn$prepare_users_target: novalue;
: 93 0420 1
: 94 0421 1 |
: 95 0422 1 | External References:
: 96 0423 1 |
: 97 0424 1 |
: 98 0425 1 external routine
: 99 0426 1     cli$get_value: addressing_mode(general),
100 0427 1     cli$present: addressing_mode(general),
101 0428 1     lib$free_vm: addressing_mode(general),
102 0429 1     lib$get_vm: addressing_mode(general),
103 0430 1     ots$cvt_ti_l: addressing_mode(general),
104 0431 1     phn$answered,
105 0432 1     phn$busy,
106 0433 1     phn$cmd_parse,
107 0434 1     phn$exit_handler,
108 0435 1     phn$facsimile2,
109 0436 1     phn$forced_link,
110 0437 1     phn$held,
111 0438 1     phn$help2,
112 0439 1     phn$hungup,
113 0440 1     phn$init_slave,
114 0441 1     phn$init_term,
115 0442 1     phn$inform,
116 0443 1     phn$kbd_get,
117 0444 1     phn$kbd_route,
118 0445 1     phn$directory2,
119 0446 1     phn$listen,
120 0447 1     phn$make_pub,
121 0448 1     phn$mbx_enable,
122 0449 1     phn$mbx_name,
123 0450 1     phn$rang_in,
124 0451 1     phn$rejected,
125 0452 1     phn$ring_out,
126 0453 1     phn$stalk,
127 0454 1     phn$unheld,
128 0455 1     str$trim: addressing_mode(general);
: 129 0456 1

```

```
: 130      0457 1 !  
: 131      0458 1 ! Own Variables:  
: 132      0459 1 !  
: 133      0460 1 !  
: 134      0461 1 ! The following is the head of the SMB queue, which is only manipulated  
: 135      0462 1 ! by this module.  
: 136      0463 1 !  
: 137      0464 1 own  
: 138      0465 1     smb_queue_head: vector[2,long]  
: 139      0466 1     initial(rep 2 of (smb_queue_head));
```

```
141 0467 1 !  
142 0468 1 ! Global Variables:  
143 0469 1 !  
144 0470 1 ! The following declarations constitute all the global data for the facility.  
145 0471 1 !  
146 0472 1 ! First we declare a buffer to hold our DECnet node name:  
147 0473 1 !  
148 0474 1 global  
149 0475 1     own_described_buffer(phn$gq_node_name,9);  
150 0476 1 !  
151 0477 1 ! Next we declare the variables that will contain the command qualifiers:  
152 0478 1 !  
153 0479 1 global  
154 0480 1     own_described_buffer(phn$gq_switch_hook,1),  
155 0481 1     phn$gl_viewport_size: long,  
156 0482 1     phn$gb_scroll: byte;  
157 0483 1 !  
158 0484 1 ! Now we declare the head of the PUB chain.  
159 0485 1 !  
160 0486 1 global  
161 0487 1     phn$gq_pubhead: vector[2,long]  
162 0488 1         initial(rep 2 of (phn$gq_pubhead));  
163 0489 1 !  
164 0490 1 ! Finally, we have a byte of global flags necessary for controlling  
165 0491 1 ! screen dynamics.  
166 0492 1 !  
167 0493 1 global  
168 0494 1     phn$gb_flags: byte  
169 0495 1         initial(%b'00000000');
```

```

171 0496 1 %sbttl 'PHN$PHONE - Mainline Routine'
172 0497 1 |++
173 0498 1 | Functional Description:
174 0499 1 | This is the main routine for the PHONE facility.
175 0500 1 |
176 0501 1 | Formal Parameters:
177 0502 1 | none
178 0503 1 |
179 0504 1 | Implicit Inputs:
180 0505 1 | global data
181 0506 1 |
182 0507 1 | Implicit Outputs:
183 0508 1 | global data
184 0509 1 |
185 0510 1 | Returned Value:
186 0511 1 | none
187 0512 1 |
188 0513 1 | Side Effects:
189 0514 1 |
190 0515 1 | --
191 0516 1 |
192 0517 1 |
193 0518 2 global routine phn$phone: novalue = begin
194 0519 2
195 0520 2 own
196 0521 2     routine_table: vector[19,long] initial(
197 0522 2         phn$kbd_get,
198 0523 2         phn$kbd_route,
199 0524 2         phn$cmd_parse,
200 0525 2         phn$talk,
201 0526 2         phn$help2,
202 0527 2         phn$ring_out,
203 0528 2         0, ! Network slave SMB only.
204 0529 2         phn$rang_in,
205 0530 2         phn$hungup,
206 0531 2         phn$busy,
207 0532 2         phn$answered,
208 0533 2         phn$rejected,
209 0534 2         0, ! Network slave SMB only.
210 0535 2         phn$listen,
211 0536 2         phn$directory2,
212 0537 2         phn$facsimile2,
213 0538 2         phn$forced_link,
214 0539 2         phn$held,
215 0540 2         phn$unheld);
216 0541 2
217 0542 2 local
218 0543 2     status: long,
219 0544 2     trnlmlst : $itmlst_decl(items=1),
220 0545 2     s: ref smb;
221 0546 2
222 0547 2
223 0548 2 ! The very first thing we do is find out if this VAX is running DECnet, and
224 0549 2 ! set up a global buffer with the node name, or null string if not.
225 0550 2 ! Eliminate any underscore in the name for prettiness.
226 0551 2
227 P 0552 2 $itmlst_init(itmlst=trnlmlst,

```



```

: 228 P 0553 2      (itmcod=lnm$string,bufadr=.phn$gq_node_name[ptr],
: 229   0554 2      bufstz=.phn$gq_node_name[len],retlen=phn$gq_node_name[len]);
: 230   0555 2
: 231 P 0556 2      status = $trnlm(attr=%ref(lnm$case blind),
: 232 P 0557 2      tabnam=$descriptor('LNM$SYSTEM'),
: 233 P 0558 2      lognam=$descriptor('SYS$NODE'),
: 234 P 0559 2      acmode=%ref(psl$exec),
: 235   0560 2      itmlst=trnlmlst);
: 236   0561 2
: 237   0562 2      if .status eqlu ss$_nolognam then
: 238   0563 2          phn$gq_node_name[len] = 0
: 239   0564 3      else (
: 240   0565 3          check (.status);
: 241   0566 4          if ch$rchar(.phn$gq_node_name[ptr]) eqlu '_' then (
: 242   0567 4              dec (phn$gq_node_name[len]);
: 243   0568 4              inc (phn$gq_node_name[ptr]);
: 244   0569 3          );
: 245   0570 2      );
: 246   0571 2
: 247   0572 2      ! It is possible that we have been fired up as a network slave for
: 248   0573 2      ! someone on a remote node that wants to talk to someone on our node.
: 249   0574 2      ! The following routine will check for this, and will never return
: 250   0575 2      ! if it is the case.
: 251   0576 2
: 252   0577 2      phn$init_slave();
: 253   0578 2
: 254   0579 2      ! I guess not. Begin an interactive session by initializing.
: 255   0580 2
: 256   0581 2      phn$init_main();
: 257   0582 2
: 258   0583 2      ! This is the main processing loop. We wait for routines to put Steering
: 259   0584 2      ! Message Blocks on the queue. We then remove them one by one and call
: 260   0585 2      ! the specified steering message routine, passing it the message text in
: 261   0586 2      ! the block. Whenever the queue empties, we go back to waiting.
: 262   0587 2
: 263   0588 3      loop (
: 264   0589 3          ! Wait for an SMB to be placed on the queue. There may already be one.
: 265   0590 3
: 266   0591 3          status = $clref(efn=phn$k_smbefn);
: 267   0592 3          if .status nequ ss$_wassef then (
: 268   0593 4              check (.status);
: 269   0594 4              status = $waitfr(efn=phn$k_smbefn);
: 270   0595 4              check (.status);
: 271   0596 4          );
: 272   0597 3      );
: 273   0598 3
: 274   0599 3      ! We have at least one SMB. Go into a loop removing them from the
: 275   0600 3      ! front of the queue. When we empty the queue, go back and wait.
: 276   0601 3
: 277   0602 4      while not remque(.smb_queue_head[0],s) do (
: 278   0603 4
: 279   0604 4          ! If the SMB type code is valid, call the appropriate
: 280   0605 4          ! steering message routine. Pass it the message text
: 281   0606 4          ! descriptor, which it can clobber.
: 282   0607 4
: 283   0608 4          if (.s[smb_w_type] gequ 1) and
: 284   0609 4              (.s[smb_w_type] lequ %allocation(routine_table)/4) then

```

```

: 285 0610 4
: 286 0611 4
: 287 0612 4
: 288 0613 4
: 289 0614 4
: 290 0615 4
: 291 0616 4
: 292 0617 4
: 293 0618 3
: 294 0619 2 );
: 295 0620 2
: 296 0621 1 end;

```

```

else (.routine_table[.s[smb_w_type]-1]) (s[smb_q_msg])
phn$inform(phn$_badsmb);

! Get rid of the SMB.
phn$kill_smb(.s);
);

```

```

.TITLE PHONE PHONE - VAX/VMS Telephone Facility
.IDENT \V04-000\

.PSECT $PLITS$,NOWRT,NOEXE,2

4D 45 54 53 59 53 24 4D 4E 4C 0000 P.AAB: .ASCII \LNMS$SYSTEM\
0000A .BLKB 2
0000000A 0000C P.AAA: .LONG 10
00000000' 00010 .ADDRESS P.AAB
45 44 4F 4E 24 53 59 53 00014 P.AAD: .ASCII \SYS$NODE\
00000008 0001C P.AAC: .LONG 8
00000000' 00020 .ADDRESS P.AAD

.PSECT $OWNS$,NOEXE,2

00000000' 00000 SMB_QUEUE_HEAD:
00000000' 00004 .ADDRESS SMB_QUEUE_HEAD
00000000G 00000000G 00000000G 00000000G 00000000G 00000000G 00008 ROUTINE_TABLE:
00000000' 00020 .ADDRESS PHN$KBD_GET, PHN$KBD_ROUTE, -
00000000G 00000000G 00000000G 00000000G 00000000G 00024 .ADDRESS PHN$CMD_PARSE, PHN$STACK, PHN$HELP2, -
00000000' 00038 .LONG 0 PHN$RING_OUT
00000000G 00000000G 00000000G 00000000G 00000000G 0003C .ADDRESS PHN$RANG_IN, PHN$HUNGUP, PHN$BUSY, -
00000000' 00018 .LONG 0 PHN$ANSWERED, PHN$REJECTED
00000000G 00000000G 00000000G 00000000G 00000000G 0001C .ADDRESS PHN$LISTEN, PHN$DIRECTORY2, -
00000000' 0001D .LONG 0 PHN$FACSIMILE2, PHN$FORCED_LINK, -
00000000G 00020 PHN$HELD, PHN$UNHELD

.PSECT $GLOBALS$,NOEXE,2

00000009 00000 PHN$GQ_NODE_NAME::
00000000' 00004 .LONG 9 .ADDRESS PHN$GQ_NODE_NAME+8
00000000' 00008 .BLKB 9
00000000' 00011 .BLKB 3
00000001 00014 PHN$GQ_SWITCH_HOOK::
00000000' 00018 .LONG 1 .ADDRESS PHN$GQ_SWITCH_HOOK+8
00000000' 0001C .BLKB 1
00000000' 0001D .BLKB 3
00000000' 00020 PHN$GL_VIEWPORT_SIZE::

```

```

00024 PHN$GB_SCROLL:: .BLKB 4
00025 .BLKB 1
00000000' 00028 PHN$GQ_PUBHEAD:: .BLKB 3
00000000' 0002C .ADDRESS PHN$GQ_PUBHEAD ;
00 00030 PHN$GB_FLAGS:: .ADDRESS PHN$GQ_PUBHEAD ;
. BYTE 0 ;

.EXTRN PHN$ OK, PHN$ ANSWERED
.EXTRN PHN$ BUSYCALL, PHN$ CANCELL
.EXTRN PHN$ CANTREACH, PHN$ CONFCALL
.EXTRN PHN$ DEAD, PHN$ DECNETLINK
.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 PHN$ IVREDUNCALL
.EXTRN PHN$ LINKERROR, PHN$ NEEDUSER
.EXTRN PHN$ NOCALL, PHN$ NOROLDS
.EXTRN PHN$ NOPORTS, PHN$ NOPRIV
.EXTRN PHN$ NOPROC, PHN$ NOTCONV
.EXTRN PHN$ ONLYNODE, PHN$ PHONEBUSY
.EXTRN PHN$ REMOTEERROR
.EXTRN PHN$ TARGTERM, PHN$ UNPLUGGED
.EXTRN PHN$ BADTERM, PHN$ SHAREDMBX
.EXTRN PHN$ INPUTTERM, CLISGET VALUE
.EXTRN CLISPRESENT, LIB$FREE VM
.EXTRN LIB$GET VM, OTSSCVT TT_L
.EXTRN PHN$ANSWERED, PHN$BUSY
.EXTRN PHN$CMD_PARSE, PHN$EXIT_HANDLER
.EXTRN PHN$FACSIMILE2, PHN$FORCED_LINK
.EXTRN PHN$HELD, PHN$HELP2
.EXTRN PHN$HUNGUP, PHN$INIT_SLAVE
.EXTRN PHN$INIT_TERM, PHN$INFORM
.EXTRN PHN$KBD_GET, PHN$KBD_ROUTE
.EXTRN PHN$DIRECTORY2, PHN$CISTEN
.EXTRN PHN$MAKE_PUB, PHN$MBX_ENABLE
.EXTRN PHN$MBX_NAME, PHN$RANG_IN
.EXTRN PHN$REJECTED, PHN$RING_OUT
.EXTRN PHN$TALK, PHN$UNHELD
.EXTRN STR$TRIM, SYS$TRNLNM
.EXTRN SYS$CLREF, SYS$WAITFR

.PSECT $CODE$,NOWRT,2

.ENTRY PHN$PHONE, Save R2,R3,R4,R5 ; 0518
MOVAB LIB$SIGNAL, R5 ;
MOVAB PHN$GQ_NODE_NAME, R4 ;

```

```

003C 00000
55 00000000G 00 9E 00002
54 0000' CF 9E 00009

```

	5E		18	C2	0000E	SUBL2	#24, SP		
	50	08	AE	9E	00011	MOVAB	TRNLNMLST, \$\$ITMBLKPTR		0554
	80		64	B0	00015	MOVW	PHN\$GQ_NODE_NAME, (\$\$ITMBLKPTR)+		
	80		02	B0	00018	MOVW	#2, (\$\$ITMBLKPTR)+		
	80	04	A4	D0	0001B	MOVL	PHN\$GQ_NODE_NAME+4, (\$\$ITMBLKPTR)+		
	80		64	9E	0001F	MOVAB	PHN\$GQ_NODE_NAME, (\$\$ITMBLKPTR)+		
			80	D4	00022	CLRL	(\$\$ITMBLKPTR)+		
		08	AE	9F	00024	PUSHAB	TRNLNMLST		0560
08	AE		01	D0	00027	MOVL	#1, 8(SP)		
		08	AE	9F	0002B	PUSHAB	8(SP)		
		0000'	CF	9F	0002E	PUSHAB	P.AAC		
		0000'	CF	9F	00032	PUSHAB	P.AAA		
10	AE	02000000	8F	D0	00036	MOVL	#33554432, 16(SP)		
		10	AE	9F	0003E	PUSHAB	16(SP)		
00000000G	00		05	FB	00041	CALLS	#5, SYS\$TRNLNM		
	53		50	D0	00048	MOVL	R0, STATUS		
000001BC	8F		53	D1	0004B	CMPL	STATUS, #444		0562
			04	12	00052	BNEQ	1\$		
			64	B4	00054	CLRW	PHN\$GQ_NODE_NAME		0563
			14	11	00056	BRB	3\$		
	05		53	E8	00058	1\$:	BLBS	STATUS, 2\$	0565
			53	DD	0005B		PUSHL	STATUS	
	65		01	FB	0005D	CALLS	#1, LIB\$SIGNAL		
5F	8F	04	B4	91	00060	2\$:	CMPB	@PHN\$GQ_NODE_NAME+4, #95	0566
			05	12	00065		BNEQ	3\$	
			64	B7	00067		DECW	PHN\$GQ_NODE_NAME	0567
		04	A4	D6	00069		INCL	PHN\$GQ_NODE_NAME+4	0568
0000G	CF		00	FB	0006C	3\$:	CALLS	#0, PHN\$INIT_SLAVE	0577
0000V	CF		00	FB	00071		CALLS	#0, PHN\$INIT_MAIN	0581
			03	DD	00076	4\$:	PUSHL	#3	0592
00000000G	00		01	FB	00078		CALLS	#1, SYSS\$CLREF	
	53		50	D0	0007F		MOVL	R0, STATUS	
	09		53	D1	00082		CMPL	STATUS, #9	0593
			1C	13	00085		BEQL	6\$	
	05		53	E8	00087		BLBS	STATUS, 5\$	0594
			53	DD	0008A		PUSHL	STATUS	
	65		01	FB	0008C		CALLS	#1, LIB\$SIGNAL	
			03	DD	0008F	5\$:	PUSHL	#3	0595
00000000G	00		01	FB	00091		CALLS	#1, SYSS\$WAITFR	
	53		50	D0	00098		MOVL	R0, STATUS	
	05		53	E8	0009B		BLBS	STATUS, 6\$	0596
			53	DD	0009E		PUSHL	STATUS	
	65		01	FB	000A0		CALLS	#1, LIB\$SIGNAL	
	52	0000'	DF	0F	000A3	6\$:	REMQUE	@SMB_QUEUE_HEAD, S	0602
			CC	1D	000A8		BVS	4\$	
		0A	A2	B5	000AA		TSTW	10(S)	0608
			18	13	000AD		BEQL	7\$	
	13	0A	A2	B1	000AF		CMPW	10(S), #19	0609
			12	1A	000B3		BGTRU	7\$	
	50	0A	A2	3C	000B5		MOVZWL	10(S), R0	0611
	50	0000'	CF	40	000B9		MOVL	ROUTINE_TABLE-4[R0], R0	
		0C	A2	9F	000BF		PUSHAB	12(S)	
	60		01	FB	000C2		CALLS	#1, (R0)	
			0B	11	000C5		BRB	8\$	
		00000000G	8F	DD	000C7	7\$:	PUSHL	#PHN\$ BADSMB	0613
0000G	CF		01	FB	000CD		CALLS	#1, PHN\$INFORM	
			52	DD	000D2	8\$:	PUSHL	S	0617

PHONE
V04-000

PHONE - VAX/VMS Telephone Facility
PHNSPHONE - Mainline Routine

M 11
16-Sep-1984 02:15:58
14-Sep-1984 12:53:28

VAX-11 Bliss-32 V4.0-742
DISK\$VMMASTER:[PHONE.SRC]PHONE.B32;1
Page 11 (4)

0000V CF

01 FB 000D4
C8 11 000D9

CALLS #1, PHNSKILL_SMB
BRB 6\$

:
: 0602

; Routine Size: 219 bytes, Routine Base: \$CODE\$ + 0000

```

: 298 0622 1 %sbttl 'PHN$INIT_MAIN - Initialize Our Wonderfullness'
: 299 0623 1 !++
: 300 0624 1 ! Functional Description:
: 301 0625 1 ! This routine is called at the very beginning of execution
: 302 0626 1 ! to initialize everything.
: 303 0627 1 !
: 304 0628 1 ! Formal Parameters:
: 305 0629 1 ! none
: 306 0630 1 !
: 307 0631 1 ! Implicit Inputs:
: 308 0632 1 ! global data
: 309 0633 1 !
: 310 0634 1 ! Implicit Outputs:
: 311 0635 1 ! global data
: 312 0636 1 !
: 313 0637 1 ! Returned Value:
: 314 0638 1 ! none
: 315 0639 1 !
: 316 0640 1 ! Side Effects:
: 317 0641 1 !
: 318 0642 1 ! --
: 319 0643 1 !
: 320 0644 1 !
: 321 0645 2 global routine phn$init_main: novalue = begin
: 322 0646 2
: 323 0647 2 own
: 324 0648 2     exit_status: long,
: 325 0649 2     exit_control_block: vector[4,long]
: 326 0650 2         initial(0,phn$exit_handler,1,exit_status);
: 327 0651 2 own
: 328 0652 2     own_described_buffer(user_name,12);
: 329 0653 2 bind
: 330 0654 2     get_name = uplit(word(12),word(jpi$username),
: 331 0655 2         long(user_name+8),
: 332 0656 2         long(user_name),
: 333 0657 2         long(0));
: 334 0658 2
: 335 0659 2 local
: 336 0660 2     status: long,
: 337 0661 2     op: ref pub;           ! Pointer to our PUB.
: 338 0662 2
: 339 0663 2
: 340 0664 2 ! We begin by obtaining our user name.
: 341 0665 2
: 342 P 0666 2 status = $getjpi(efn=phn$k_getjpiefn,
: 343 0667 2     itmlst=get_name);
: 344 0668 2 check (.status);
: 345 0669 2 status = $waitfr(efn=phn$k_getjpiefn);
: 346 0670 2 check (.status);
: 347 0671 2 str$trim(user_name,user_name,user_name);
: 348 0672 2
: 349 0673 2 ! Now we can create a PUB to represent ourselves, known as 'our PUB'. Our
: 350 0674 2 ! PUB is always assumed to be the first one on the PUB chain.
: 351 0675 2
: 352 0676 2 status = phn$make_pub(user_name,op);
: 353 0677 2 check (.status);
: 354 0678 2 op[pub_v_temporary] = false;

```

```
355 0679 2
356 0680 2 ! Now we have to assign ourselves to our receive mailbox, the mailbox that
357 0681 2 ! other users will send us messages in. The mailbox may already have been
358 0682 2 ! created by some other user trying to call us, so first we try to assign
359 0683 2 ! to it.
360 0684 2
361 0685 3 begin
362 0686 3 local
363 0687 3     local_described_buffer(mbx_name,4+32);
364 0688 3
365 0689 3 phn$mbx_name(user_name,mbx_name);
366 P 0690 3 status = $assign(devnam=mbx_name,
367 0691 3     chan=op[pub_w_channel]);
368 0692 4 if (.status nequ ss$_nosuchdev)
369 0693 3     and (.status nequ ss$_ivdevnam) then
370 0694 4     check (.status)
371 0695 4 else (
372 0696 4     ! Nope, our mailbox doesn't exist. Try to create a permanent
373 0697 4     ! mailbox with the name. Mark it for deletion so we don't leave
374 0698 4     ! crud around later.
375 0699 4
376 P 0700 4     status = $crembx(prmflg=1,
377 P 0701 4         chan=op[pub_w_channel],
378 P 0702 4         maxmsg=phn$_mbxsize,
379 0703 4         lognam=mbx_name);
380 0704 5     if .status nequ ss$_nopriv then (
381 0705 5         check (.status);
382 0706 5         status = $delmbx(chan=.op[pub_w_channel]);
383 0707 5         check (.status);
384 0708 5     ) else (
385 0709 5
386 0710 5         ! Too bad, we don't have the privilege to create a permanent
387 0711 5         ! mailbox. Try to create a temporary one so the guy can at
388 0712 5         ! least talk to him/herself.
389 0713 5
390 P 0714 5         status = $crembx(prmflg=0,
391 P 0715 5             chan=op[pub_w_channel],
392 P 0716 5             maxmsg=phn$_mbxsize,
393 0717 5             lognam=mbx_name);
394 0718 5         check (.status);
395 0719 4     );
396 0720 3 );
397 0721 2 end;
398 0722 2
399 0723 2 ! Now we can enable our mailbox, allowing an AST to be delivered when someone
400 0724 2 ! sends us a message. There may already be a message waiting.
401 0725 2
402 0726 2 phn$mbx_enable();
403 0727 2
404 0728 2 ! Now we have to obtain the various goodies from the command line. This is
405 0729 2 ! done after enabling our mailbox so any messages from other people will be
406 0730 2 ! handled before any subcommand included on the PHONE command.
407 0731 2
408 0732 2 phn$dcl_command_line();
409 0733 2
410 0734 2 ! Finally we initialize the user's terminal. This is done last so that
411 0735 2 ! anything the user types will be handled after any subcommand included
```

```

: 412      0736 2 ! on the PHONE command.
: 413      0737 2
: 414      0738 2 status = phn$init_term();
: 415      0739 2 check (.status);
: 416      0740 2
: 417      0741 2 ! From this point on, we want to perform some clean-up if the user exits.
: 418      0742 2
: 419      0743 2 status = $dclexh(desblk=exit_control_block);
: 420      0744 2 check (.status);
: 421      0745 2
: 422      0746 2 return;
: 423      0747 2
: 424      0748 1 end;

```

```

.PSECT $SPLITS,NOWRT,NOEXE,2
      000C 00024 P.AAE: .WORD 12
      0202 00026 .WORD 514
00000000' 00028 .ADDRESS USER_NAME+8
00000000' 0002C .ADDRESS USER_NAME
00000000 00030 .LONG 0

```

```

.PSECT $OWNS,NOEXE,2
      00054 EXIT_STATUS:
      00000000 00058 EXIT_CONTROL_BLOCK:
      00000000G 0005C .BLKB 4
      00000001 00060 .LONG 0
      00000000' 00064 .ADDRESS PHN$EXIT_HANDLER
      0000000C 00068 .LONG 1
      00000000' 0006C .ADDRESS EXIT_STATUS
      00000000' 00070 .ADDRESS USER_NAME+8
      00068 USER_NAME:
      0006C .LONG 12
      00070 .ADDRESS USER_NAME+8
      00070 .BLKB 12

```

```

GET_NAME= P.AAE
.EXTRN SYSS$GETJPI, SYSS$ASSIGN
.EXTRN SYSS$CREMBX, SYSS$DELMBX
.EXTRN SYSS$DCLEXH

```

```

.PSECT $CODE$,NOWRT,2
      007C 00000 .ENTRY PHN$INIT MAIN, Save R2,R3,R4,R5,R6
56 00000000G 00 9E 00002 MOVAB SYSS$CREMBX, R6
55 0000' CF 9E 00009 MOVAB USER_NAME, R5
54 00000000G 00 9E 0000E MOVAB LIB$SIGNAL, R4
5E 30 C2 00015 SUBL2 #48, SP
7E 7C 00018 CLRQ -(SP)
7E D4 0001A CLRL -(SP)
0000' CF 9F 0001C PUSHAB GET_NAME
7E 7C 00020 CLRQ -(SP)
01 DD 00022 PUSHL #1
00000000G 00 07 FB 00024 CALLS #7, SYSS$GETJPI
53 50 D0 0002B MOVL R0, STATUS

```


	05		53	E8	0002E	BLBS	STATUS, 1\$		0668
			53	DD	00031	PUSHL	STATUS		
	64		01	FB	00033	CALLS	#1, LIB\$SIGNAL		
			01	DD	00036	PUSHL	#1		0669
00000000G	00		01	FB	00038	CALLS	#1, SYSS\$WAITFR		
	53		50	DD	0003F	MOVL	R0, STATUS		
	05		53	E8	00042	BLBS	STATUS, 2\$		0670
			53	DD	00045	PUSHL	STATUS		
	64		01	FB	00047	CALLS	#1, LIB\$SIGNAL		
			55	DD	0004A	PUSHL	R5		0671
			55	DD	0004C	PUSHL	R5		
			55	DD	0004E	PUSHL	R5		
00000000G	00		03	FB	00050	CALLS	#3, STR\$TRIM		
		4020	8F	BB	00057	PUSHR	#^M<R5, SP>		0676
0000G	CF		02	FB	0005B	CALLS	#2, PHN\$MAKE_PUB		
	53		50	DD	00060	MOVL	R0, STATUS		
	05		53	E8	00063	BLBS	STATUS, 3\$		0677
			53	DD	00066	PUSHL	STATUS		
	64		01	FB	00068	CALLS	#1, LIB\$SIGNAL		
	52		6E	DD	0006B	MOVL	OP, R2		0678
00F0	C2		04	8A	0006E	BICB2	#4, 240(R2)		
	04		24	DD	00073	MOVL	#36, MBX_NAME		0687
	08		AE	9E	00077	MOVAB	MBX_NAME+8, MBX_NAME+4		
		0C	AE	9F	0007C	PUSHAB	MBX_NAME		0689
		04	55	DD	0007F	PUSHL	R5		
0000G	CF		02	FB	00081	CALLS	#2, PHN\$MBX_NAME		
			7E	7C	00086	CLRQ	-(SP)		0691
	52	00F4	C2	9E	00088	MOVAB	244(R2), R2		
			52	DD	0008D	PUSHL	R2		
		10	AE	9F	0008F	PUSHAB	MBX_NAME		
00000000G	00		04	FB	00092	CALLS	#4, SYSS\$ASSIGN		
	53		50	DD	00099	MOVL	R0, STATUS		
00000908	8F		53	D1	0009C	CMPL	STATUS, #2>12		0692
			09	13	000A3	BEQL	4\$		
00000144	8F		53	D1	000A5	CMPL	STATUS, #324		0693
			45	12	000AC	BNEQ	8\$		
		04	AE	9F	000AE	PUSHAB	MBX_NAME		0703
			7E	7C	000B1	CLRQ	-(SP)		
			7E	D4	000B3	CLRL	-(SP)		
	7E	0100	8F	3C	000B5	MOVZWL	#256, -(SP)		
			52	DD	000BA	PUSHL	R2		
			01	DD	000BC	PUSHL	#1		
	66		07	FB	000BE	CALLS	#7, SYSS\$CREMBX		
	53		50	DD	000C1	MOVL	R0, STATUS		
	24		53	D1	000C4	CMPL	STATUS, #36		0704
			14	13	000C7	BEQL	6\$		
	05		53	E8	000C9	BLBS	STATUS, 5\$		0705
			53	DD	000CC	PUSHL	STATUS		
	64		01	FB	000CE	CALLS	#1, LIB\$SIGNAL		
	7E		62	3C	000D1	MOVZWL	(R2), -(SP)		0706
00000000G	00		01	FB	000D4	CALLS	#1, SYSS\$DELMBX		
			13	11	000DB	BRB	7\$		
		04	AE	9F	000DD	PUSHAB	MBX_NAME		0717
			7E	7C	000E0	CLRQ	-(SP)		
			7E	D4	000E2	CLRL	-(SP)		
	7E	0100	8F	3C	000E4	MOVZWL	#256, -(SP)		
			52	DD	000E9	PUSHL	R2		

		7E	D4	000EB		CLRL	-(SP)	:
	66	07	FB	000ED		CALLS	#7, SYSSCREMBX	:
	53	50	DO	000F0	7\$:	MOVL	R0, STATUS	:
	05	53	E8	000F3	8\$:	BLBS	STATUS, 9\$: 0718
		53	DD	000F6		PUSHL	STATUS	:
	64	01	FB	000F8		CALLS	#1, LIB\$SIGNAL	:
0000G	CF	00	FB	000FB	9\$:	CALLS	#0, PHNSMBX_ENABLE	: 0726
0000V	CF	00	FB	00100		CALLS	#0, PHNSDCL_COMMAND_LINE	: 0732
0000G	CF	00	FB	00105		CALLS	#0, PHNSINIT_TERM	: 0738
	53	50	DO	0010A		MOVL	R0, STATUS	:
	05	53	E8	0010D		BLBS	STATUS, 10\$: 0739
		53	DD	00110		PUSHL	STATUS	:
	64	01	FB	00112		CALLS	#1, LIB\$SIGNAL	:
		A5	9F	00115	10\$:	PUSHAB	EXIT CONTROL BLOCK	: 0743
00000000G	00	01	FB	00118		CALLS	#1, SYSSDCLXH	:
	53	50	DO	0011F		MOVL	R0, STATUS	:
	05	53	E8	00122		BLBS	STATUS, 11\$: 0744
		53	DD	00125		PUSHL	STATUS	:
	64	01	FB	00127		CALLS	#1, LIB\$SIGNAL	:
		04	0012A	11\$:		RET		: 0748

; Routine Size: 299 bytes, Routine Base: \$CODE\$ + 00DB

```

: 426 0749 1 %sbttl 'PHN$DCL_COMMAND_LINE - Get Command Line Goodies'
: 427 0750 1 !++
: 428 0751 1 ! Functional Description:
: 429 0752 1 ! This routine is called to 'parse the command line'. All the work
: 430 0753 1 ! has been done by DCL, but we need to use the callback mechanism
: 431 0754 1 ! to get the information.
: 432 0755 1 !
: 433 0756 1 ! Formal Parameters:
: 434 0757 1 ! none
: 435 0758 1 !
: 436 0759 1 ! Implicit Inputs:
: 437 0760 1 ! global data
: 438 0761 1 !
: 439 0762 1 ! Implicit Outputs:
: 440 0763 1 ! global data
: 441 0764 1 !
: 442 0765 1 ! Returned Value:
: 443 0766 1 ! none
: 444 0767 1 !
: 445 0768 1 ! Side Effects:
: 446 0769 1 !
: 447 0770 1 ! --
: 448 0771 1 !
: 449 0772 1 !
: 450 0773 2 global routine phn$dcl_command_line: novalue = begin
: 451 0774 2
: 452 0775 2 local
: 453 0776 2 status: long;
: 454 0777 2
: 455 0778 2
: 456 0779 2 ! First we get the switch hook character as specified by the qualifier.
: 457 0780 2
: 458 0781 2 cli$get_value(describe('SWITCH_HOOK'),phn$gq_switch_hook);
: 459 0782 2
: 460 0783 2 ! Now we get the viewport size, as specified by the qualifier. We have
: 461 0784 2 ! to convert the value to binary and make sure it's within the valid range.
: 462 0785 2
: 463 0786 3 begin
: 464 0787 3 local
: 465 0788 3 local_described_buffer(size_buf,10);
: 466 0789 3
: 467 0790 3 cli$get_value(describe('VIEWPORT_SIZE'),size_buf);
: 468 0791 3 status = ots$cvt_ti_l(size_buf,phn$gl_viewport_size,4,%b'11');
: 469 0792 3 if .status eglu $$$_normal then
: 470 0793 3 phn$gl_viewport_size = min(max(pub_k_minlines,.phn$gl_viewport_size),
: 471 0794 3 pub_k_maxlines)
: 472 0795 3 else
: 473 0796 3 phn$gl_viewport_size = pub_k_maxlines;
: 474 0797 2 end;
: 475 0798 2
: 476 0799 2 ! Now we get the scroll flag, as specified by the qualifier.
: 477 0800 2
: 478 0801 2 phn$gb_scroll = cli$present(describe('SCROLL'));
: 479 0802 2
: 480 0803 2 ! Now we have to retrieve the command line tokens. If present, these
: 481 0804 2 ! make up a phone command that we are to execute first. Sit in a loop
: 482 0805 2 ! and queue cmd_parse steering messages for each token.
```

```

: 483 0806 2
: 484 0807 2 incru i from 1 to 4 do (
: 485 0808 3
: 486 0809 3 local
: 487 0810 3 local_described_buffer(token_buf,80);
: 488 0811 3
: 489 0812 4 cli$get_value((case .i from 1 to 4 of set
: 490 0813 4 [1]: describe('TOKEN1');
: 491 0814 4 [2]: describe('TOKEN2');
: 492 0815 4 [3]: describe('TOKEN3');
: 493 0816 4 [4]: describe('TOKEN4');
: 494 0817 4 tes); token_buf);
: 495 0818 3 str$trim(token_buf,token_buf,token_buf);
: 496 0819 3
: 497 0820 4 if .token_buf[len] gequ 1 then (
: 498 0821 4 phn$queue_smb(smb__cmd_parse,describe(' '));
: 499 0822 4 phn$queue_smb(smb__cmd_parse,token_buf);
: 500 0823 3 );
: 501 0824 2 );
: 502 0825 2
: 503 0826 2 ! Now we can queue a carriage return to end the command. It doesn't hurt
: 504 0827 2 ! if there wasn't a command.
: 505 0828 2
: 506 0829 2 phn$queue_smb(smb__cmd_parse,describe(%char(ret)));
: 507 0830 2
: 508 0831 2 return;
: 509 0832 2
: 510 0833 1 end;

```

											.PSECT		\$SPLITS,NOWRT,NOEXE,2					
	4B	4F	4F	48	5F	48	43	54	49	57	53	00034	P.AAG:	.ASCII	\SWITCH_HOOK\	:		
												0003F		.BLKB	1	:		
												0000000B	00040	P.AAF:	.LONG	11	:	
												00000000	00044		.ADDRESS	P.AAG	:	
45	5A	49	53	5F	54	52	4F	50	57	45	49	56	00048	P.AAI:	.ASCII	\VIEWPORT_SIZE\	:	
													00055		.BLKB	3	:	
													0000000D	00058	P.AAH:	.LONG	13	:
													00000000	0005C		.ADDRESS	P.AAI	:
						4C	4C	4F	52	43	53	00060	P.AAK:	.ASCII	\SCROLL\	:		
													00066		.BLKB	2	:	
													00000006	00068	P.AAJ:	.LONG	6	:
													00000000	0006C		.ADDRESS	P.AAK	:
						31	4E	45	4B	4F	54	00070	P.AAM:	.ASCII	\TOKEN1\	:		
													00076		.BLKB	2	:	
													00000006	00078	P.AAL:	.LONG	6	:
													00000000	0007C		.ADDRESS	P.AAM	:
						32	4E	45	4B	4F	54	00080	P.AAO:	.ASCII	\TOKEN2\	:		
													00086		.BLKB	2	:	
													00000006	00088	P.AAN:	.LONG	6	:
													00000000	0008C		.ADDRESS	P.AAO	:
						33	4E	45	4B	4F	54	00090	P.AAQ:	.ASCII	\TOKEN3\	:		
													00096		.BLKB	2	:	
													00000006	00098	P.AAP:	.LONG	6	:
													00000000	0009C		.ADDRESS	P.AAQ	:

```

34 4E 45 4B 4F 54 000A0 P.AAS: .ASCII \TOKEN4\
                                000A6 .BLKB 2
                                000A8 P.AAR: .LONG 6
00000006 000AC .ADDRESS P.AAS
00000000' 000B0 P.AAU: .ASCII \ \
                                000B1 .BLKB 3
                                000B4 P.AAT: .LONG 1
00000001 000B8 .ADDRESS P.AAU
00000000' 000BC P.AAW: .ASCII <13>
                                000BD .BLKB 3
                                000C0 P.AAV: .LONG 1
00000001 000C4 .ADDRESS P.AAW
00000000'

```

.PSECT \$CODE\$,NOWRT,2

```

007C 00000 .ENTRY PHN$DCL COMMAND LINE, Save R2,R3,R4,R5,R6 : 0773
56 0000V CF 9E 00002 MOVAB PHN$QUEUE SMB, R6
55 0000' CF 9E 00007 MOVAB PHN$GL_VIEWPORT_SIZE, R5
54 00000000G 00 9E 0000C MOVAB CLISGET_VALJE, R4
53 0000' CF 9E 00013 MOVAB P.AAF, R3
5E A8 AE 9E 00018 MOVAB -88(SP), SP
F4 A5 9F 0001C PUSHAB PHN$GQ_SWITCH_HOOK : 0781
53 DD 0001F PUSHL R3
64 02 FB 00021 CALLS #2, CLISGET_VALUE
44 AE 0A D0 00024 MOVL #10, SIZE_BUF : 0788
48 AE 4C AE 9E 00028 MOVAB SIZE_BUF+8, SIZE_BUF+4
44 AE 44 AE 9F 0002D PUSHAB SIZE_BUF : 0790
18 A3 9F 00030 PUSHAB P.AAR
64 02 FB 00033 CALLS #2, CLISGET_VALUE
03 DD 00036 PUSHL #3 : 0791
04 DD 00038 PUSHL #4
55 DD 0003A PUSHL R5
50 AE 9F 0003C PUSHAB SIZE_BUF
00000000G 00 04 FB 0003F CALLS #4, OTSS$CVT_TI_L
01 50 D1 00046 CMPL STATUS, #1 : 0792
18 12 00049 BNEQ 3$
50 65 D0 0004B MOVL PHN$GL_VIEWPORT_SIZE, R0 : 0793
03 50 D1 0004E CMPL R0, #3
03 18 00051 BGEQ 1$
50 03 D0 00053 MOVL #3, R0
0A 50 D1 00056 1$: CMPL R0, #10
03 15 00059 BLEQ 2$
50 0A D0 0005B MOVL #10, R0
65 50 D0 0005E 2$: MOVL R0, PHN$GL_VIEWPORT_SIZE
03 11 00061 BRB 4$
65 0A D0 00063 3$: MOVL #10, PHN$GL_VIEWPORT_SIZE : 0796
00000000G 28 A3 9F 00066 4$: PUSHAB P.AAJ : 0801
00 01 FB 00069 CALLS #1, CLISPRESENT
04 A5 50 90 00070 MOVAB R0, PHN$GB_SCROLL
52 01 D0 00074 MOVL #1, I : 0807
6E 50 8F 9A 00077 5$: MOVZBL #80, TOKEN_BUF : 0810
04 AE 08 AE 9E 0007B MOVAB TOKEN_BUF+8, TOKEN_BUF+4
5E DD 00080 PUSHL SP : 0812
001A 03 01 52 CF 00082 CASEL I, #1, #3
0014 000E 0008 00086 6$: .WORD 7$-6$,-

```

PHONE
V04-000

PHONE - VAX/VMS Telephone Facility
PHNSDCL_COMMAND_LINE - Get Command

I 12
16-Sep-1984 02:15:58
Line Goodies 14-Sep-1984 12:53:28

VAX-11 Bliss-32 V4.0-742
DISK\$VMMASTER:[PHONE.SRC]PHONE.B32;1

Page 20
(6)

```

      8$-6$,-
      9$-6$,-
      10$-6$
50     38   A3  9E 0008E 7$:   MOVAB  P.AAL, R0      : 0813
      10   11 00092         BRB      11$
50     48   A3  9E 00094 8$:   MOVAB  P.AAN, R0      : 0814
      0A   11 00098         BRB      11$
50     58   A3  9E 0009A 9$:   MOVAB  P.AAP, R0      : 0815
      04   11 0009E         BRB      11$
50     68   A3  9E 000A0 10$:  MOVAB  P.AAR, R0      : 0816
      50   DD 000A4 11$:     PUSHL   R0
64     02   FB 000A6         CALLS   #2, CLISGET_VALUE : 0812
      5E   DD 000A9         PUSHL   SP              : 0818
      04   AE  9F 000AB         PUSHAB  TOKEN_BUF
      08   AE  9F 000AE         PUSHAB  TOKEN_BUF
00000000G 00   03   FB 000B1         CALLS   #3, STRSTRIM
      6E   B5 000B8         TSTW   TOKEN_BUF      : 0820
      0F   13 000BA         BEQL   12$
      74   A3  9F 000BC         PUSHAB  P.AAT          : 0821
      03   DD 000BF         PUSHL   #3
66     02   FB 000C1         CALLS   #2, PHNSQUEUE_SMB : 0822
      5E   DD 000C4         PUSHL   SP
      03   DD 000C6         PUSHL   #3
66     02   FB 000C8         CALLS   #2, PHNSQUEUE_SMB : 0807
      52   D6 000CB 12$:     INCL   I
04     52   D1 000CD         CMLL   I, #4
      A5   1B 000D0         BLEQU  5$
      0080 C3   9F 000D2         PUSHAB  P.AAV          : 0829
      03   DD 000D6         PUSHL   #3
66     02   FB 000D8         CALLS   #2, PHNSQUEUE_SMB : 0833
      04   000DB         RET

```

; Routine Size: 220 bytes, Routine Base: \$CODE\$ + 0206

```

512 0834 1 %sbttl 'PHN$QUEUE_SMB - Queue a Steering Message Block'
513 0835 1 ++
514 0836 1 Functional Description:
515 0837 1 This routine is called to create a Steering Message Block and
516 0838 1 queue in onto the SMB queue.
517 0839 1
518 0840 1 Formal Parameters:
519 0841 1 type The type code for the SMB.
520 0842 1 text Address of descriptor of optional message text.
521 0843 1
522 0844 1 Implicit Inputs:
523 0845 1 global data
524 0846 1
525 0847 1 Implicit Outputs:
526 0848 1 global data
527 0849 1
528 0850 1 Returned Value:
529 0851 1 none
530 0852 1
531 0853 1 Side Effects:
532 0854 1
533 0855 1 --
534 0856 1
535 0857 1
536 0858 2 global routine phn$queue_smb(type,text): novalue = begin
537 0859 2
538 0860 2 bind
539 0861 2 text_dsc = .text: descriptor;
540 0862 2
541 0863 2 local
542 0864 2 status: long,
543 0865 2 smb_size: long,
544 0866 2 s: ref smb;
545 0867 2
546 0868 2 builtin
547 0869 2 nullparameter;
548 0870 2
549 0871 2
550 0872 2 ! We begin by calculating the total size of the SMB and allocating
551 0873 2 ! memory for it.
552 0874 2
553 0875 3 smb_size = smb_k_size + (if nullparameter(2) then 0
554 0876 2 else .text_dsc[len]);
555 0877 2 status = lib$get_vm(smb_size,s);
556 0878 2 check (.status);
557 0879 2
558 0880 2 ! Now we fill in the new SMB with its length, the message type code,
559 0881 2 ! and the message text. We also build a descriptor for the message text.
560 0882 2
561 0883 2 s[smb_w_length] = .smb_size;
562 0884 2 s[smb_w_type] = .type;
563 0885 3 begin
564 0886 3 bind
565 0887 3 smb_msg_dsc = s[smb_q_msg]: descriptor;
566 0888 3
567 0889 3 if nullparameter(2) then
568 0890 3 smb_msg_dsc[len] = 0

```

```

: 569 0891 4 else (
: 570 0892 4     ch$move(.text_dsc[len],.text_dsc[ptr],s[smb_t_msgbuf]);
: 571 0893 4     smb_msg_dsc[len] = .text_dsc[len];
: 572 0894 4     smb_msg_dsc[ptr] = s[smb_t_msgbuf];
: 573 0895 3 );
: 574 0896 2 end;
: 575 0897 2
: 576 0898 2 ! Now we can queue the new SMB onto the end of the steering message queue.
: 577 0899 2 ! If the queue was empty, make sure we set the event flag to awaken the
: 578 0900 2 ! main loop.
: 579 0901 2
: 580 0902 2 status = insque(.s,.smb_queue_head[1]);
: 581 0903 2 if .status eqlu 1 then
: 582 0904 2     $setef(efn=phn$k_smbefn);
: 583 0905 2
: 584 0906 2 return;
: 585 0907 2
: 586 0908 1 end;

```

.EXTRN SYS\$SETEF

			03FC 00000	.ENTRY PHN\$QUEUE_SMB, Save R2,R3,R4,R5,R6,R7,R8,R9 ;	0858
	5E		08 C2 00002	SUBL2 #8, SP	
	58	08	AC D0 00005	MOVL TEXT, R8	0861
	02		6C 91 00009	CMPB (AP), #2	0875
			05 1F 0000C	BLSSU 1\$	
		08	AC D5 0000E	TSTL 8(AP)	
			04 12 00011	BNEQ 2\$	
			50 D4 00013 1\$:	CLRL R0	
			03 11 00015	BRB 3\$	
	50		68 3C 00017 2\$:	MOVZWL (R8), R0	0876
	04	AE	14 A0 9E 0001A 3\$:	MOVAB 20(R0), SMB_SIZE	0875
			5E DD 0001F	PUSHL SP	0877
		08	AE 9F 00021	PUSHAB SMB_SIZE	
	00000000G	00	02 FB 00024	CALLS #2, LIB\$GET_VM	
		59	50 D0 0002B	MOVL R0, STATUS	
		09	59 E8 0002E	BLBS STATUS, 4\$	0878
			59 DD 00031	PUSHL STATUS	
	00000000G	00	01 FB 00033	CALLS #1, LIB\$SIGNAL	
		57	6E D0 0003A 4\$:	MOVL S, R7	0883
	08	A7	04 AE B0 0003D	MOVW SMB_SIZE, 8(R7)	
	0A	A7	04 AC B0 00042	MOVW TYPE, 10(R7)	0884
		56	0C A7 9E 00047	MOVAB 12(R7), R6	0887
		02	6C 91 0004B	CMPB (AP), #2	0889
			05 1F 0004E	BLSSU 5\$	
		08	AC D5 00050	TSTL 8(AP)	
			04 12 00053	BNEQ 6\$	
			66 B4 00055 5\$:	CLRW (R6)	0890
			0E 11 00057	BRB 7\$	
	14	A7	04 B8 28 00059 6\$:	MOV C3 (R8), @4(R8), 20(R7)	0892
		66	68 B0 0005F	MOVW (R8), (R6)	0893
		04	A7 9E 00062	MOVAB 20(R7), 4(R6)	0894
			50 D4 00067 7\$:	CLRL R0	0902
	0000'	DF	67 0E 00069	INSQUE (R7), @SMB_QUEUE_HEAD+4	
			02 12 0006E	BNEQ 8\$	

PHONE
V04-000

PHONE - VAX/VMS Telephone Facility
PHN\$QUEUE_UMB - Queue a Steering Message Block

L 12
16-Sep-1984 02:15:58
14-Sep-1984 12:53:28

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

Page 23
(7)

		50	D6	00070		INCL	R0	:	
	59	50	D0	00072	8\$:	MOVL	R0, STATUS	:	
	01	59	D1	00075		CMPL	STATUS, #1	:	0903
		09	12	00078		BNEQ	9\$:	
		03	DD	0007A		PUSHL	#3	:	0904
00000000G	00	01	FB	0007C		CALLS	#1, SYS\$SETEF	:	
		04	00083	9\$:	RET			:	0908

; Routine Size: 132 bytes, Routine Base: \$CODE\$ + 02E2

```

: 588 0909 1 %sbttl 'PHN$KILL_SMB - Kill an Obsolete SMB'
: 589 0910 1 ++
: 590 0911 1 Functional Description:
: 591 0912 1 This routine is called to deallocate an obsolete Steering
: 592 0913 1 Message Block.
: 593 0914 1
: 594 0915 1 Formal Parameters:
: 595 0916 1 obsolete_smb Address of the obsolete SMB.
: 596 0917 1
: 597 0918 1 Implicit Inputs:
: 598 0919 1 global data
: 599 0920 1
: 600 0921 1 Implicit Outputs:
: 601 0922 1 global data
: 602 0923 1
: 603 0924 1 Returned Value:
: 604 0925 1 none
: 605 0926 1
: 606 0927 1 Side Effects:
: 607 0928 1
: 608 0929 1 --
: 609 0930 1
: 610 0931 1
: 611 0932 2 global routine phn$kill_smb(obsolete_smb): novalue = begin
: 612 0933 2
: 613 0934 2 bind
: 614 0935 2 os = .obsolete_smb: smb;
: 615 0936 2
: 616 0937 2 local
: 617 0938 2 status: long,
: 618 0939 2 smb_size: long;
: 619 0940 2
: 620 0941 2
: 621 0942 2 ! All we have to do is get the length of the SMB and deallocate it.
: 622 0943 2
: 623 0944 2 smb_size = .os[smb_w_length];
: 624 0945 2 status = lib$free_vm(smb_size,obsolete_smb);
: 625 0946 2 check (.status);
: 626 0947 2 return;
: 627 0948 2
: 628 0949 1 end;

```

```

: 0932
: 0935
: 0944
: 0945
: 0946
: 0949

```

		0000 0000	.ENTRY	PHN\$KILL_SMB, Save nothing	: 0932
50	04	AC D0 00002	MOVL	OBSOLETE_SMB, R0	: 0935
7E	08	A0 3C 00006	MOVZWL	8(R0), SMB_SIZE	: 0944
	04	AC 9F 0000A	PUSHAB	OBSOLETE_SMB	: 0945
	04	AE 9F 0000D	PUSHAB	SMB_SIZE	: 0945
00000000G 00		02 FB 00010	CALLS	#2, LIB\$FREE_VM	: 0946
09		50 E8 00017	BLBS	STATUS, 1\$: 0946
		50 DD 0001A	PUSHL	STATUS	: 0946
00000000G 00		01 FB 0001C	CALLS	#1, LIB\$SIGNAL	: 0949
		04 00023 1\$:	RET		: 0949

PHONE
V04-000

PHONE - VAX/VMS Telephone Facility
PHNSKILL_SMB - Kill an Obsolete SMB

N 12
16-Sep-1984 02:15:58
14-Sep-1984 12:53:28

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[PHONE.SRC]PHONE.B32;1
Page 25
(8)

; Routine Size: 36 bytes, Routine Base: \$CODE\$ + 0366

```

630 0950 1 %sbttl 'ANL$PREPARE_USERS_TARGET - Prepare Target Entered by User'
631 0951 1 |++
632 0952 1 | Functional Description:
633 0953 1 | This routine is called to prepare a target entered by the user.
634 0954 1 | It performs a logical name translation in case the user entered
635 0955 1 | a logical name. It also handles node names without the double colon.
636 0956 1 |
637 0957 1 | Formal Parameters:
638 0958 1 | users Address of descriptor of user's target.
639 0959 1 | just_node A flag, true if target is just a node name.
640 0960 1 | final Address of descriptor of buffer to receive final
641 0961 1 | target. We set the length.
642 0962 1 |
643 0963 1 | Implicit Inputs:
644 0964 1 | global data
645 0965 1 |
646 0966 1 | Implicit Outputs:
647 0967 1 | global data
648 0968 1 |
649 0969 1 | Returned Value:
650 0970 1 | none
651 0971 1 |
652 0972 1 | Side Effects:
653 0973 1 |
654 0974 1 | --
655 0975 1 |
656 0976 1 |
657 0977 2 global routine phn$prepare_users_target(users,just_node,final): novalue = begin
658 0978 2
659 0979 2 bind
660 0980 2 users_dsc = .users: descriptor,
661 0981 2 final_dsc = .final: descriptor;
662 0982 2
663 0983 2 local
664 0984 2 status: long;
665 0985 2
666 0986 2
667 0987 2 ! First we try translating the user's target as if it were a logical name.
668 0988 2 ! If that fails, we just move the target into the final buffer.
669 0989 2
670 P 0990 2 status = $trnlog(lognam=users_dsc,
671 P 0991 2 rslen=final_dsc[len],
672 0992 2 rslbuf=final_dsc);
673 0993 3 if .status eqv ss$_ivlognam then (
674 0994 3 final_dsc[len] = .users_dsc[len];
675 0995 3 ch$move(.users_dsc[len],.users_dsc[ptr], .final_dsc[ptr]);
676 0996 2 ) else
677 0997 2 check (.status);
678 0998 2
679 0999 2 ! Now if this is supposed to be just a node name, the double colon is
680 1000 2 ! optional. Add it if we need to.
681 1001 2
682 1002 2 if .just_node and .final_dsc[len] gtru 0 then
683 1003 3 if ch$neq(2,.final_dsc[ptr]+.final_dsc[len]-2, 2,uplit byte ('::'),' ') then (
684 1004 3 ch$move(2,uplit byte ('::'), .final_dsc[ptr]+.final_dsc[len]);
685 1005 3 final_dsc[len] = .final_dsc[len] + 2;
686 1006 2 );

```

PHONE
V04-000

PHONE - VAX/VMS Telephone Facility
ANL\$PREPARE_USERS_TARGET - Prepare Target Enter

C 13
16-Sep-1984 02:15:58
14-Sep-1984 12:53:28

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[PHONE.SRC]PHONE.B32;1 Page 27
(9)

PU
VO

: 687
: 688
: 689
: 690
1007 2
1008 2 return;
1009 2
1010 1 end;

1007 2
1008 2 return;
1009 2
1010 1 end;

.PSECT \$SPLITS,NOWRT,NOEXE,2

3A 3A 000C8 P.AAX: .ASCII \::\
3A 3A 000CA P.AAY: .ASCII \::\
:

.EXTRN SYS\$TRNLOG

.PSECT \$CODE\$,NOWRT,2

007C 00000 .ENTRY PHN\$PREPARE_USERS_TARGET, Save R2,R3,R4,R5,-; 0977

			52	04	AC	D0	00002	MOVL	USERS, R2	0980	
			56	0C	AC	D0	00006	MOVL	FINAL, R6	0981	
					7E	7C	0000A	CLRQ	-(SP)	0992	
					7E	D4	0000C	CLRL	-(SP)		
					56	DD	0000E	PUSHL	R6		
				0044	8F	BB	00010	PUSHR	#^M<R2,R6>		
		00000000G	00		06	FB	00014	CALLS	#6, SYS\$TRNLOG		
		00000154	8F		50	D1	0001B	CMPL	STATUS, #340	0993	
					0B	12	00022	BNEQ	1\$		
			66		62	B0	00024	MOVW	(R2), (R6)	0994	
	04	B6	04	B2	62	28	00027	MOVW3	(R2), @4(R2), @4(R6)	0995	
					0C	11	0002D	BRB	2\$	0993	
			09		50	E8	0002F	1\$:	BLBS	STATUS, 2\$	0997
					50	DD	00032	PUSHL	STATUS		
		00000000G	00		01	FB	00034	CALLS	#1, LIB\$SIGNAL		
			1B	08	AC	E9	0003B	2\$:	BLBC	JUST_NODE, 3\$	1002
					66	B5	0003F	TSTW	(R6)		
					17	13	00041	BEQL	3\$		
			50		66	3C	00043	MOVZWL	(R6), R0	1003	
			50	04	A6	C0	00046	ADDL2	4(R6), R0		
		0000'	CF	FE	A0	B1	0004A	CMPW	-2(R0), P.AAX		
					08	13	00050	BEQL	3\$		
			60	0000'	CF	B0	00052	MOVW	P.AAY, (R0)	1004	
			66		02	A0	00057	ADDW2	#2, (R6)	1005	
					04	0005A	3\$:	RET		1010	

: Routine Size: 91 bytes, Routine Base: \$CODE\$ + 038A

: 691
: 692
1011 1
1012 0 end eludom

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

PHONE
V04-000

PHONE - VAX/VMS Telephone Facility
ANL\$PREPARE_USERS_TARGET - Prepare Target Enter

D 13
16-Sep-1984 02:15:58
14-Sep-1984 12:53:28

VAX-11 Bliss-32 v4.0-742
DISK\$VMSMASTER:[PHONE.SRC]PHONE.B32;1 Page 28 (9)

Name	Bytes	Attributes
\$OWNS	124	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$GLOBAL\$	49	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$SPLITS	204	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODE\$	997	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	35	0	581	00:00.7

COMMAND QUALIFIERS

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

Size: 997 code + 377 data bytes
 Run Time: 00:14.9
 Elapsed Time: 00:59.4
 Lines/CPU Min: 4077
 Lexemes/CPU-Min: 33215
 Memory Used: 156 pages
 Compilation Complete

