

```

PPPPPPPPPPPP    HHH      HHH      000000000    NNN      NNN      EEEEEEEEEEEEEEE
PPPPPPPPPPPPPP  HHH      HHH      000000000    NNN      NNN      EEEEEEEEEEEEEEE
PPPPPPPPPPPPPP  HHH      HHH      000000000    NNN      NNN      EEEEEEEEEEEEEEE
PPP              PPP  HHH      HHH      000          000    NNN      NNN      EEE
PPP              PPP  HHH      HHH      000          000    NNN      NNN      EEE
PPP              PPP  HHH      HHH      000          000    NNN      NNN      EEE
PPP              PPP  HHH      HHH      000          000    NNNNNN      NNN      EEE
PPP              PPP  HHH      HHH      000          000    NNNNNN      NNN      EEE
PPP              PPP  HHH      HHH      000          000    NNNNNN      NNN      EEE
PPPPPPPPPPPPPP  HHHHHHHHHHHHHHHHHHH  000          000    NNN      NNN      EEEEEEEEEEEE
PPPPPPPPPPPPPP  HHHHHHHHHHHHHHHHHHH  000          000    NNN      NNN      EEEEEEEEEEEE
PPPPPPPPPPPPPP  HHHHHHHHHHHHHHHHHHH  000          000    NNN      NNN      EEEEEEEEEEEE
PPP              HHH      HHH      000          000    NNN      NNNNNN      EEE
PPP              HHH      HHH      000          000    NNN      NNNNNN      EEE
PPP              HHH      HHH      000          000    NNN      NNNNNN      EEE
PPP              HHH      HHH      000          000    NNN      NNN      EEE
PPP              HHH      HHH      000          000    NNN      NNN      EEE
PPP              HHH      HHH      000          000    NNN      NNN      EEE
PPP              HHH      HHH      000          000    NNN      NNN      EEE
PPP              HHH      HHH      000000000    NNN      NNN      EEEEEEEEEEEEEEE
PPP              HHH      HHH      000000000    NNN      NNN      EEEEEEEEEEEEEEE
PPP              HHH      HHH      000000000    NNN      NNN      EEEEEEEEEEEEEEE

```

[illegible]

```

SSSSSSSS TTTTTTTTT AAAA AA CCCCCCCC KK KK CCCCCCCC MM MM DDDDDDDD SSSSSSSS
SSSSSSSS TTTTTTTTT AAAA AA CCCCCCCC KK KK CCCCCCCC MM MM DDDDDDDD SSSSSSSS
SS      TT      AA      AA CC      KK      KK CC      MMMM      MM      DD      DD SS
SS      TT      AA      AA CC      KK      KK CC      MMMM      MM      DD      DD SS
SS      TT      AA      AA CC      KK      KK CC      MM      MM      DD      DD SS
SS      TT      AA      AA CC      KK      KK CC      MM      MM      DD      DD SS
      SSSSSS      TT      AA      AA CC      KKKKKK      CC      MM      MM      DD      DD SSSSSS
      SSSSSS      TT      AA      AA CC      KKKKKK      CC      MM      MM      DD      DD SSSSSS
      SS      TT      AA      AA CC      KK      KK CC      MM      MM      DD      DD SS
      SS      TT      AA      AA CC      KK      KK CC      MM      MM      DD      DD SS
      SS      TT      AA      AA CC      KK      KK CC      MM      MM      DD      DD SS
      SS      TT      AA      AA CC      KK      KK CC      MM      MM      DD      DD SS
SSSSSSSS TT      AA      AA CCCCCCCC KK      KK CCCCCCCC MM      MM DDDDDDDD SSSSSSSS
SSSSSSSS TT      AA      AA CCCCCCCC KK      KK CCCCCCCC MM      MM DDDDDDDD 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

```



```
0001 0 %title 'STACKCMDS - Stacking Commands'
0002 0      module stackcmds (
0003 1      ident='V04-000') = begin
0004 1
0005 1
0006 1 *****
0007 1 *
0008 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0009 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0010 1 *  ALL RIGHTS RESERVED.
0011 1 *
0012 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0013 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0014 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0015 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0016 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0017 1 *  TRANSFERRED.
0018 1 *
0019 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0020 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0021 1 *  CORPORATION.
0022 1 *
0023 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0024 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0025 1 *
0026 1 *
0027 1 *****
0028 1
0029 1
0030 1 ++
0031 1 Facility:      VAX/VMS Telephone Facility, Stacking Commands
0032 1
0033 1 Abstract:      This module handles the following PHONE commands:
0034 1                  HOLD      Put people on hold.
0035 1                  UNHOLD    Take people off of hold.
0036 1
0037 1
0038 1 Environment:
0039 1
0040 1 Author: Paul C. Anagnostopoulos, Creation Date: 11 November 1980
0041 1
0042 1 Modified By:
0043 1
0044 1      V03-001 PCA0047      Paul Anagnostopoulos      26-Mar-1982
0045 1      Minor changes to convert from process name to user name.
0046 1 --
```



```
: 48      0047 1 %sbttl 'Module Declarations'
: 49      0048 1
: 50      0049 1  Libraries and Requires:
: 51      0050 1
: 52      0051 1
: 53      0052 1  library 'sys$library:starlet.l32';
: 54      0053 1  require 'phonereq';
: 55      0382 1
: 56      0383 1
: 57      0384 1  Table of Contents:
: 58      0385 1
: 59      0386 1
: 60      0387 1  forward routine
: 61      0388 1      phn$hold_cmd: novalue,
: 62      0389 1      phn$held: novalue,
: 63      0390 1      phn$unhold_cmd: novalue,
: 64      0391 1      phn$unheld: novalue;
: 65      0392 1
: 66      0393 1
: 67      0394 1  External References:
: 68      0395 1
: 69      0396 1
: 70      0397 1  external routine
: 71      0398 1      phn$break_link,
: 72      0399 1      phn$cmp_target,
: 73      0400 1      phn$conversing,
: 74      0401 1      phn$fresh_hold,
: 75      0402 1      phn$fresh_screen,
: 76      0403 1      phn$inform,
: 77      0404 1      phn$make_tsb,
: 78      0405 1      phn$send_smb;
: 79      0406 1
: 80      0407 1
: 81      0408 1  Own Variables:
: 82      0409 1
```



```
84 0410 1 %sbttl 'PHN$HOLD_CMD - Handle HOLD Command'
85 0411 1 ++
86 0412 1 Functional Description:
87 0413 1 This routine handles the HOLD command. This command is used to
88 0414 1 put all of the people currently engaged in conversation on hold.
89 0415 1
90 0416 1 Formal Parameters:
91 0417 1 none
92 0418 1
93 0419 1 Implicit Inputs:
94 0420 1 global data
95 0421 1
96 0422 1 Implicit Outputs:
97 0423 1 global data
98 0424 1
99 0425 1 Returned Value:
100 0426 1 none
101 0427 1
102 0428 1 Side Effects:
103 0429 1
104 0430 1 --
105 0431 1
106 0432 1
107 0433 2 global routine phn$hold_cmd: novalue = begin
108 0434 2
109 0435 2 local
110 0436 2 p: ref pub;
111 0437 2
112 0438 2
113 0439 2 ! First we check to see if we're talking to someone. If not, that's an error.
114 0440 2
115 0441 3 if not phn$conversing() then (
116 0442 3 phn$inform(phn$_notconv);
117 0443 3 return;
118 0444 2 );
119 0445 2
120 0446 2 ! How we scan the PUB chain and adjust the hold status of each person.
121 0447 2 There are two cases:
122 0448 2 1. The person is already on hold. We decrement their hold depth
123 0449 2 so that they are on a deeper hold.
124 0450 2 2. We are talking to the person or they have us on hold.
125 0451 2 Now they are on hold, and we have to send them a message to
126 0452 2 that effect. Also update the viewport flags.
127 0453 2
128 0454 2 p = .phn$gq_pubhead[0];
129 0455 3 until .p eqla phn$gq_pubhead do (
130 0456 3 if not .p[pub_v_temporary] then
131 0457 4 if dec (p[pub_w_depth]) eql -1 then (
132 0458 4 phn$send_smb(.p,smb_held);
133 0459 4 p[pub_v_uhaveheld] = true;
134 0460 4 phn$fresh_hold(.p);
135 0461 3 );
136 0462 3 p = .p[pub_l_flink];
137 0463 2 );
138 0464 2
139 0465 2 return;
140 0466 2
```

STACKCMLS
V04-000

STACKCMLS - Stacking Commands
PHN\$HOLD_CMD - Handle HOLD Command

: 141

0467 1 end;

I 15
16-Sep-1984 02:18:10
14-Sep-1984 12:53:32

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

Page 4
(3)

.TITLE STACKCMLS STACKCMLS - Stacking Commands
.IDENT \V04-000\

.EXTRN PHN\$ _OK, PHN\$ _ANSWERED
.EXTRN PHN\$ _BUSYCALL, PHN\$ _CANCALL
.EXTRN PHN\$ _CANTREACH, PHN\$ _CONFCALL
.EXTRN PHN\$ _DEAD, PHN\$ _DECNETLINK
.EXTRN PHN\$ _DIRCAN, PHN\$ _FACSCAN
.EXTRN PHN\$ _HELPCAN, PHN\$ _HUNGUP
.EXTRN PHN\$ _JISTRANG, 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\$ _NOHOLDS
.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, PHN\$ _GQ_NODE_NAME
.EXTRN PHN\$ _GQ_SWITCH_HOOK
.EXTRN PHN\$ _GL_VIEWPORT_SIZE
.EXTRN PHN\$ _GB_SCROLL, PHN\$ _GQ_PUBHEAD
.EXTRN PHN\$ _GB_FLAGS, PHN\$ _BREAK_LINK
.EXTRN PHN\$ _CMP_TARGET, PHN\$ _CONVERSING
.EXTRN PHN\$ _FRESH_HOLD, PHN\$ _FRESH_SCREEN
.EXTRN PHN\$ _INFORM, PHN\$ _MAKE_TSB
.EXTRN PHN\$ _SEND_SMB

.PSECT \$CODE\$,NOWRT,2

.ENTRY PHN\$HOLD_CMD, Save R2
CALLS #0, PHN\$CONVERSING
BLBS R0, 1\$
PUSHL #PHN\$NOTCONV
CALLS #1, PHN\$INFORM
RET
MOVL @PHN\$GQ_PUBHEAD, P
MOVAB PHN\$GQ_PUBHEAD, R0
CMPL P, R0
BEQL 4\$
BBS #2, 240(P), 3\$
CVTBL 242(P), R0
DECL R0
MOVW R0, 242(P)
CMPL R0, #-1

0000G CF 0004 00000
OC 00 FB 00002
0000G CF 00000000G 50 E8 00007
01 FB 0000A
04 00015
52 0000G DF D0 00016 1\$:
50 0000G CF 9E 0001B 2\$:
50 52 D1 00020
35 13 00023
2A 00F0 C2 02 E0 00025
50 00F2 C2 32 0002B
50 D7 00030
00F2 C2 50 B0 00032
FFFFFFFF 8F 50 D1 00037

: 0433
: 0441
: 0442
: 0441
: 0454
: 0455
: 0456
: 0457
:

STACKCMDS
V04-000

STACKCMDS - Stacking Commands
PHN\$HOLD_CMD - Handle HOLD Command

J 15
16-Sep-1984 02:18:10
14-Sep-1984 12:53:32

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

Page 5
(3)

0000G CF
00F0 C2

0000G CF
52

15 12 0003E
12 DD 00040
52 DD 00042
02 FB 00044
01 88 00049
52 DD 0004E
01 FB 00050
62 D0 00055 3\$:
C1 11 00058
04 0005A 4\$:

BNEQ 3\$
PUSHL #18
PUSHL P
CALLS #2, PHN\$SEND_SMB
BISB2 #1, 240(P)
PUSHL P
CALLS #1, PHN\$FRESH_HOLD
MOVL (P), P
BRB 2\$
RET

: 0458
: 0459
: 0460
: 0462
: 0455
: 0467

; Routine Size: 91 bytes, Routine Base: \$CODE\$ + 0000

```
143 0468 1 %sbttl 'PHN$HELD - Handle Held Steering Message'
144 0469 1 ++
145 0470 1 Functional Description:
146 0471 1 This steering message routine handles the held message, which someone
147 0472 1 sends us when they put us on hold. We have to record the fact
148 0473 1 that we are on hold.
149 0474 1
150 0475 1 Formal Parameters:
151 0476 1 from_msg The address of a descriptor of the node/user name
152 0477 1 spec of the person putting us on hold. It is
153 0478 1 followed by an eofrom character.
154 0479 1
155 0480 1 Implicit Inputs:
156 0481 1 global data
157 0482 1
158 0483 1 Implicit Outputs:
159 0484 1 global data
160 0485 1
161 0486 1 Returned Value:
162 0487 1 none
163 0488 1
164 0489 1 Side Effects:
165 0490 1
166 0491 1 --
167 0492 1
168 0493 1
169 0494 2 global routine phn$held(from_msg): novalue = begin
170 0495 2
171 0496 2 bind
172 0497 2 from_msg_dsc = .from_msg: descriptor;
173 0498 2
174 0499 2 local
175 0500 2 status: long,
176 0501 2 from_tsb: tsb,
177 0502 2 p: ref pub;
178 0503 2
179 0504 2
180 0505 2 ! We begin by adjusting the message descriptor to exclude the eofrom character
181 0506 2 ! and making a TSB to parse the person's spec.
182 0507 2
183 0508 2 dec (from_msg_dsc[len]);
184 0509 2 status = phn$make_tsb(from_msg_dsc,from_tsb);
185 0510 2 check (.status);
186 0511 2
187 0512 2 ! Now we scan the PUB chain, looking for the PUB representing this person.
188 0513 2 ! If we find one, we set the "they have us held" flag.
189 0514 2
190 0515 2 p = .phn$gq_pubhead[0];
191 0516 3 until .p eqla phn$gq_pubhead do (
192 0517 3 if (not .p[pub_v_temporary]) and
193 0518 3 (not .p[pub_v_hasuheld]) then
194 0519 3
195 0520 4 if phn$cmp_target(p[pub_b_tsb],from_tsb) then (
196 0521 4 p[pub_v_hasuheld] = true;
197 0522 4 phn$fresh_hold(.p);
198 0523 4 );
199 0524 3
```


STACKCMDS
V04-000

STACKCMDS - Stacking Commands
PHN\$HELD - Handle Held Steering Message

L 15
16-Sep-1984 02:18:10
14-Sep-1984 12:53:32

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

Page 7
(4)

```
: 200      0525 3      p = .p[pub_l_flink];
: 201      0526 2 );
: 202      0527 2
: 203      0528 2 return;
: 204      0529 2
: 205      0530 1 end;
```

				000C 00000	.ENTRY PHN\$HELD, Save R2,R3	: 0494
	5E	FF1C	CE	9E 00002	MOVAB -228(SP), SP	: 0508
		04	BC	B7 00007	DECW @FROM_MSG	: 0509
			5E	DD 0000A	PUSHL SP	
		04	AC	DD 0000C	PUSHL FROM MSG	
0000G	CF		02	FB 0000F	CALLS #2, PHN\$MAKE_TSB	
	09		50	E8 00014	BLBS STATUS, 1\$	: 0510
			50	DD 00017	PUSHL STATUS	
00000000G	00		01	FB 00019	CALLS #1, LIB\$SIGNAL	
	52	0000G	DF	D0 00020 1\$:	MOVL @PHN\$GQ_PUBHEAD, P	: 0515
	50	0000G	CF	9E 00025 2\$:	MOVAB PHN\$GQ_PUBHEAD, R0	: 0516
	50		52	D1 0002A	CMPL P, R0	
			29	13 0002D	BEQL 4\$	
	53	00F0	C2	9E 0002F	MOVAB 240(P), R3	: 0517
1B	63		02	E0 00034	BBS #2, (R3), 3\$	
17	63		01	E0 00038	BBS #1, (R3), 3\$	: 0518
			5E	DD 0003C	PUSHL SP	: 0520
		0C	A2	9F 0003E	PUSHAB 12(P)	
0000G	CF		02	FB 00041	CALLS #2, PHN\$CMP_TARGET	
	0A		50	E9 00046	BLBC R0, 3\$	
	63		02	88 00049	BISB2 #2, (R3)	: 0521
			52	DD 0004C	PUSHL P	: 0522
0000G	CF		01	FB 0004E	CALLS #1, PHN\$FRESH_HOLD	
	52		62	D0 00053 3\$:	MOVL (P), P	: 0525
			CD	11 00056	BRB 2\$	: 0516
			04	00058 4\$:	RET	: 0530

; Routine Size: 89 bytes, Routine Base: \$CODE\$ + 005B

```
207 0531 1 %sbtll 'PHN$UNHOLD_CMD - Handle UNHOLD Command'
208 0532 1 ++
209 0533 1 Functional Description:
210 0534 1 This routine handles the UNHOLD command, which is used to bring
211 0535 1 people off of hold. The hold stack is effectively popped, bringing
212 0536 1 the last group of held people back to life. People currently in
213 0537 1 the conversation are hung up on.
214 0538 1
215 0539 1 Formal Parameters:
216 0540 1 none
217 0541 1
218 0542 1 Implicit Inputs:
219 0543 1 global data
220 0544 1
221 0545 1 Implicit Outputs:
222 0546 1 global data
223 0547 1
224 0548 1 Returned Value:
225 0549 1 none
226 0550 1
227 0551 1 Side Effects:
228 0552 1
229 0553 1 --
230 0554 1
231 0555 1
232 0556 2 global routine phn$unhold_cmd: novalue = begin
233 0557 2
234 0558 2 local
235 0559 2 unholdable: byte,
236 0560 2 p: ref pub, p2: ref pub;
237 0561 2
238 0562 2
239 0563 2 ! We want to make sure that there is at least one person on hold.
240 0564 2 ! Otherwise the command should be ignored. Just scan the PUB chain
241 0565 2 ! looking for someone on hold.
242 0566 2
243 0567 2 unholdable = false;
244 0568 2 p = ..phn$gq_pubhead[0];
245 0569 3 until .p eqla phn$gq_pubhead do (
246 0570 4 if not .p[pub_v_temporary] and .p[pub_v_uhaveheld] then (
247 0571 4 unholdable = true;
248 0572 4 exitloop;
249 0573 3 );
250 0574 3 p = .p[pub_l_flink];
251 0575 3 );
252 0576 2 );
253 0577 2
254 0578 2 ! If no one was on hold, tell the user and forget it.
255 0579 2
256 0580 3 if not .unholdable then (
257 0581 3 phn$inform(phn$_noholds);
258 0582 3 return;
259 0583 2 );
260 0584 2
261 0585 2 ! Now we scan the PUB chain and act on each PUB. There are three cases:
262 0586 2 ! 1. The person is down two or more hold levels. The person
263 0587 2 ! remains on hold, but at a shallower depth.
```



```
: 264 0588 2 !      2.      The person is down one hold level.  They are brought off of
: 265 0589 2 !      hold status.
: 266 0590 2 !      3.      The person is currently being talked to.  They are hung up on.
: 267 0591 2
: 268 0592 2 p = .phn$gq_pubhead[0];
: 269 0593 3 until .p eqla phn$gq_pubhead do (
: 270 0594 3     p2 = .p[pub_[_flink];          ! In case we delete PUB.
: 271 0595 3
: 272 0596 3     if not .p[pub_v_temporary] then
: 273 0597 3         select one inc (p[pub_w_depth]) of set
: 274 0598 4         [0]:          (phn$send_smb(.p,smb_unheld);
: 275 0599 3                     p[pub_v_uhaveheld] = false;);
: 276 0600 3
: 277 0601 3         [1]:          phn$break_link(.p,smb_hungup);
: 278 0602 3         tes;
: 279 0603 3
: 280 0604 3     p = .p2;
: 281 0605 2 );
: 282 0606 2
: 283 0607 2 ! We have to recalculate the viewports and refresh the screen.
: 284 0608 2
: 285 0609 2 phn$fresh_screen(false);
: 286 0610 2
: 287 0611 2 return;
: 288 0612 2
: 289 0613 1 end;
```

				001C 00000	.ENTRY PHN\$UNHOLD_CMD, Save R2,R3,R4	: 0556
	54	0000G	CF	9E 00002	MOVAB PHN\$GQ_PUBHEAD, R4	
			51	94 00007	CLRB UNHOLDABLE	: 0567
	52	00	B4	D0 00009	MOVL @PHN\$GQ_PUBHEAD, P	: 0568
	50		64	9E 0000D 1\$:	MOVAB PHN\$GQ_PUBHEAD, R0	: 0569
	50		52	D1 00010	CMPL P, R0	
			15	13 00013	BEQL 3\$	
0A	00F0	C2	02	E0 00015	BBS #2, 240(P), 2\$	: 0570
		05	C2	E9 0001B	BLBC 240(P), 2\$	
		51	01	90 00020	MOVAB #1, UNHOLDABLE	: 0571
			05	11 00023	BRB 3\$	: 0570
		52	62	D0 00025 2\$:	MOVL (P), P	: 0575
			E3	11 00028	BRB 1\$	: 0569
	0C		51	E8 0002A 3\$:	BLBS UNHOLDABLE, 4\$	: 0580
		00000000G	8F	DD 0002D	PUSHL #PHN\$ NOHOLDS	: 0581
	0000G	CF	01	FB 00033	CALLS #1, PHN\$INFORM	
				04 00038	RET	: 0580
	52	00	B4	D0 00039 4\$:	MOVL @PHN\$GQ_PUBHEAD, P	: 0592
	50		64	9E 0003D 5\$:	MOVAB PHN\$GQ_PUBHEAD, R0	: 0593
	50		52	D1 00040	CMPL P, R0	
			3C	13 00043	BEQL 8\$	
	53		62	D0 00045	MOVL (P), P2	: 0594
2E	00F0	C2	02	E0 00048	BBS #2, 240(P), 7\$	: 0596
		50	C2	32 0004E	CVTDL 242(P), R0	: 0597
			50	D6 00053	INCL R0	
	00F2	C2	50	B0 00055	MOVW R0, 242(P)	

STACKCMDS
V04-000

STACKCMDS - Stacking Commands
PHN\$UNHOLD_CMD - Handle UNHOLD Command

B 16
16-Sep-1984 02:18:10
14-Sep-1984 12:53:32

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

Page 10
(5)

		50	D5	0005A	TSTL	R0		: 0598
		10	12	0005C	BNEQ	6\$		
		13	DD	0005E	PUSHL	#19		
		52	DD	00060	PUSHL	P		
0000G	CF	02	FB	00062	CALLS	#2, PHN\$SEND_SMB		
00FO	C2	01	8A	00067	BICB2	#1, 240(P)		: 0599
		0E	11	0006C	BRB	7\$		: 0597
	01	50	D1	0006E	6\$: CMPL	R0, #1		: 0601
		09	12	00071	BNEQ	7\$		
		09	DD	00073	PUSHL	#9		
		52	DD	00075	PUSHL	P		
0000G	CF	02	FB	00077	CALLS	#2, PHN\$BREAK_LINK		
	52	53	D0	0007C	7\$: MOVL	P2, P		: 0604
		BC	11	0007F	BRB	5\$		: 0593
		7E	D4	00081	8\$: CLRL	-(SP)		: 0609
0000G	CF	01	FB	00083	CALLS	#1, PHN\$FRESH_SCREEN		
		04	00088	RET				: 0613

; Routine Size: 137 bytes, Routine Base: \$CODE\$ + 00B4


```

: 291 0614 1 %sbtll 'PHN$UNHELD - Handle Unheld Message'
: 292 0615 1 ++
: 293 0616 1 Functional Description:
: 294 0617 1 This steering message routine handles the unheld message, which is sent
: 295 0618 1 to us when someone brings us off hold. We have to clear the flag
: 296 0619 1 that says we are held.
: 297 0620 1
: 298 0621 1 Formal Parameters:
: 299 0622 1 from_msg Address of descriptor of string containing
: 300 0623 1 the node/user name of the sender. It is
: 301 0624 1 followed by an eofrom character.
: 302 0625 1
: 303 0626 1 Implicit Inputs:
: 304 0627 1 global data
: 305 0628 1
: 306 0629 1 Implicit Outputs:
: 307 0630 1 global data
: 308 0631 1
: 309 0632 1 Returned Value:
: 310 0633 1 none
: 311 0634 1
: 312 0635 1 Side Effects:
: 313 0636 1
: 314 0637 1 --
: 315 0638 1
: 316 0639 1
: 317 0640 2 global routine phn$unheld(from_msg): novalue = begin
: 318 0641 2
: 319 0642 2 bind
: 320 0643 2 from_msg_dsc = .from_msg: descriptor;
: 321 0644 2
: 322 0645 2 local
: 323 0646 2 status: long,
: 324 0647 2 from_tsb: tsb,
: 325 0648 2 p: ref pub;
: 326 0649 2
: 327 0650 2
: 328 0651 2 ! We begin by adjusting the message descriptor to exclude the eofrom character
: 329 0652 2 ! and making a TSB to parse the person's spec.
: 330 0653 2
: 331 0654 2 dec (from_msg_dsc[len]);
: 332 0655 2 status = phn$make_tsb(from_msg_dsc,from_tsb);
: 333 0656 2 check (.status);
: 334 0657 2
: 335 0658 2 ! Now we scan the PUB chain, looking for the PUB representing this person.
: 336 0659 2 ! If we find one, we clear the "they have us held" flag.
: 337 0660 2
: 338 0661 2 p = ..phn$gq_pubhead[0];
: 339 0662 3 until .p eqla phn$gq_pubhead do (
: 340 0663 3 if (not .p[pub_v_temporary]) and
: 341 0664 3 .p[pub_v_hasuheld] then
: 342 0665 3
: 343 0666 4 if phn$cmp_target(p[pub_b_tsb],from_tsb) then (
: 344 0667 4 p[pub_v_hasuheld] = false;
: 345 0668 4 phn$fresh_hold(.p);
: 346 0669 3 );
: 347 0670 3 p = .p[pub_l_flink];
```


STACKCMTS
V04-000

STACKCMTS - Stacking Commands
PHN\$UNHELD - Handle Unheld Message

D 16
16-Sep-1984 02:18:10
14-Sep-1984 12:53:32

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

Page 12
(6)

```
: 348
: 349
: 350
: 351
: 352
0671 2 );
0672 2
0673 2 return;
0674 2
0675 1 end;
```

				000C 00000	.ENTRY PHN\$UNHELD, Save R2,R3	: 0640
	5E	FF1C	CE	9E 00002	MOVAB -228(SP), SP	: 0654
		04	BC	B7 00007	DECW @FROM_MSG	: 0655
			5E	DD 0000A	PUSHL SP	: 0656
		04	AC	DD 0000C	PUSHL FROM MSG	: 0661
	0000G	CF	02	FB 0000F	CALLS #2, PHN\$MAKE_TSB	: 0662
		09	50	E8 00014	BLBS STATUS, 1\$	: 0663
			50	DD 00017	PUSHL STATUS	: 0664
	00000000G	00	01	FB 00019	CALLS #1, LIB\$SIGNAL	: 0666
		52	0000G	DF D0 00020 1\$:	MOVL @PHN\$GQ_PUBHEAD, P	: 0667
		50	0000G	CF 9E 00025 2\$:	MOVAB PHN\$GQ_PUBHEAD, R0	: 0668
		50	52	D1 0002A	CMPL P, R0	: 0670
			29	13 0002D	BEQL 4\$	: 0662
		53	00F0	C2 9E 0002F	MOVAB 240(P), R3	: 0675
1B		63	02	E0 00034	BBS #2, (R3), 3\$	
17		63	01	E1 00038	BBC #1, (R3), 3\$	
			5E	DD 0003C	PUSHL SP	
			A2	9F 0003E	PUSHAB 12(P)	
	0000G	CF	02	FB 00041	CALLS #2, PHN\$CMP_TARGET	
		0A	50	E9 00046	BLBC R0, 3\$	
		63	02	8A 00049	BICB2 #2, (R3)	
			52	DD 0004C	PUSHL P	
	0000G	CF	01	FB 0004E	CALLS #1, PHN\$FRESH_HOLD	
		52	62	D0 00053 3\$:	MOVL (P), P	
			CD	11 00056	BRB 2\$	
			04	00058 4\$:	RET	

; Routine Size: 89 bytes, Routine Base: \$CODE\$ + 013D

```
: 353
: 354
0676 1
0677 0 end eludom
```

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

:	Name	Bytes	Attributes
:	\$CODE\$	406	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)

STACKCMDS
V04-000

STACKCMDS - Stacking Commands
PHN\$UNHELD - Handle Unheld Message

E 16
16-Sep-1984 02:18:10
14-Sep-1984 12:53:32

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

Page 13
(6)

Library Statistics

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

COMMAND QUALIFIERS

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

; Size: 406 code + 0 data bytes
; Run Time: 00:07.2
; Elapsed Time: 00:27.2
; Lines/CPU Min: 5618
; Lexemes/CPU-Min: 27543
; Memory Used: 93 pages
; Compilation Complete

0305 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PHONE
LIS

NETSLAVE
LIS

LINKSUBS
LIS

PHONMSG
LIS

STACKMDS
LIS

TERMINAL
LIS

MISCMD
LIS

PUBSUBS
LIS

INPUT
LIS