

DDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL

```

SSSSSSSS TTTT TTTT AAAA AA CCCCCC LL IIIII NN NN TTTT TTTT
SSSSSSSS TTTT TTTT AAAA AA CCCCCC LL IIIII NN NN TTTT TTTT
SS TT TT AA AA CC LL IIIII NN NN TTTT TTTT
SS TT TT AA AA CC LL IIIII NN NN TTTT TTTT
SS TT TT AA AA CC LL IIIII NN NN TTTT TTTT
SSSSSS TT TT AA AA CCCCCC LL IIIII NN NN TTTT TTTT
SSSSSS TT TT AA AA CCCCCC LL IIIII NN NN TTTT TTTT
SS TT TT AA AA CCCCCC LL IIIII NN NN TTTT TTTT
SS TT TT AA AA CCCCCC LL IIIII NN NN TTTT TTTT
SS TT TT AA AA CCCCCC LL IIIII NN NN TTTT TTTT
SS TT TT AA AA CCCCCC LL IIIII NN NN TTTT TTTT
SSSSSS TT TT AA AA CCCCCC LL IIIII NN NN TTTT TTTT
SSSSSS TT TT AA AA CCCCCC LL IIIII NN NN TTTT TTTT

```

```

LL IIIII SSSSSSS
LL IIIII SSSSSSS
LL II
LL II
LL II
LL II
LL II
LL II
LL II
LL II
LL II
LL IIIII SSSSSSS
LLLLLLLLLLLL IIIII SSSSSSS
LLLLLLLLLLLL IIIII SSSSSSS

```

```
1 0001 0 MODULE staclint (IDENT='V04-000',
2 0002 0 ADDRESSING_MODE(NONEXTERNAL=LONG RELATIVE,
3 0003 0 EXTERNAL=GENERAL)) =
4 0004 1 BEGIN
5 0005 1
6 0006 1
7 0007 1 *****
8 0008 1 *
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
10 0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
11 0011 1 * ALL RIGHTS RESERVED. *
12 0012 1 *
13 0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
14 0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
15 0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
16 0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
17 0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
18 0018 1 * TRANSFERRED. *
19 0019 1 *
20 0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
21 0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
22 0022 1 * CORPORATION. *
23 0023 1 *
24 0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
25 0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
26 0026 1 *
27 0027 1 *
28 0028 1 *****
29 0029 1
30 0030 1
31 0031 1 **
32 0032 1 FACILITY: Stand-alone command language interface routines
33 0033 1
34 0034 1 ABSTRACT:
35 0035 1
36 0036 1 These routines are used to enable a stand-alone
37 0037 1 image to obtain the command parameters and qualifiers
38 0038 1 from the command language interpreter.
39 0039 1
40 0040 1 ENVIRONMENT:
41 0041 1
42 0042 1 VAX/VMS operating system. unprivileged user mode,
43 0043 1
44 0044 1 AUTHOR: Peter George, October 1981
45 0045 1
46 0046 1 Modified by:
47 0047 1
48 0048 1 V03-006 PCG0006 Peter George 17-Feb-1983
49 0049 1 Include INTDEF instead of CLIDEF.
50 0050 1
51 0051 1 V03-005 PCG0005 Peter George 14-Dec-1982
52 0052 1 Allocate local descriptor for command string.
53 0053 1
54 0054 1 V03-004 PCG0004 Peter George 29-Nov-1982
55 0055 1 Add argument to str$analyze_sdesc call.
56 0056 1
57 0057 1 V03-003 PCG0003 Peter George 02-Nov-1982
```

```
: 58      0058 1  | Add CLISNEXT QUAL. Fix optional length assignment
: 59      0059 1  | in CLISGET_VALUE. Analyze input string descriptors.
: 60      0060 1  |
: 61      0061 1  | V03-002 PCG0002      Peter George      18-Oct-1982
: 62      0062 1  | Add prompt argument to CLISDCL_PARSE.
: 63      0063 1  |
: 64      0064 1  | V03-001 PCG0001      Peter George      17-Sep-1982
: 65      0065 1  | Use INT data structure.
: 66      0066 1  | Add prompt and continuation routines to CLISDCL_PARSE.
: 67      0067 1  | Add value address to CLISGET_VALUE.
: 68      0068 1  | Add user argument to CLISDISPATCH.
: 69      0069 1  | --
: 70      0070 1  |
: 71      0071 1  |
: 72      0072 1  | Include files
: 73      0073 1  |
: 74      0074 1  | LIBRARY 'SYSSLIBRARY:STARLET';
: 75      0075 1  |
: 76      0076 1  | REQUIRE 'SYSSLIBRARY:UTILDEF';      ! Common VMS BLISS definitions
: 77      0252 1  |
: 78      0253 1  | REQUIRE 'LIBS:INTDEF';              ! Interface definitions
```

80	0279	1	:		
81	0280	1	:	Table of contents	
82	0281	1	:		
83	0282	1	:		
84	0283	1	:	FORWARD ROUTINE	
85	0284	1	:	cli\$present,	! Determine if entity present
86	0285	1	:	cli\$get_value,	! Get value of entity
87	0286	1	:	cli\$dcl_parse,	! Parse a command line
88	0287	1	:	cli\$dispatch,	! Dispatch to user processing routine
89	0288	1	:	cli\$end_parse,	! Signal any unprocessed modifiers
90	0289	1	:	cli\$next_qual;	! Get next qualifier
91	0290	1	:		
92	0291	1	:	EXTERNAL ROUTINE	
93	0292	1	:	dcl\$present,	
94	0293	1	:	dcl\$getvalue,	
95	0294	1	:	dcl\$dclparse,	
96	0295	1	:	dcl\$dispatch,	
97	0296	1	:	dcl\$nextqual,	
98	0297	1	:	dcl\$endparse,	
99	0298	1	:	str\$analyze_sdesc,	! Analyze input descriptor
100	0299	1	:	lib\$get_vm,	! Allocate virtual memory
101	0300	1	:	lib\$free_vm,	! Deallocate virtual memory
102	0301	1	:	str\$copy_dx;	! Copy to any class string
103	0302	1	:		
104	0303	1	:		
105	0304	1	:	External message definitions	
106	0305	1	:		
107	0306	1	:		
108	0307	1	:	EXTERNAL LITERAL	
109	0308	1	:	cli\$_nocomd,	! Prompt was ctrl/z-ed
110	0309	1	:	cli\$_concat,	! Value is concatenated
111	0310	1	:	cli\$_comma,	! Value is comma separated
112	0311	1	:	cli\$_present,	! Entity is explicitly globally present
113	0312	1	:	cli\$_negated,	! Entity is explicitly not globally present
114	0313	1	:	cli\$_locpres,	! Entity is explicitly locally present
115	0314	1	:	cli\$_locneg,	! Entity is explicitly not locally present
116	0315	1	:	cli\$_defaulted,	! Entity is implicitly present
117	0316	1	:	cli\$_absent;	! Entity is implicitly not present

```

119 0317 1 GLOBAL ROUTINE cli$present (name) =
120 0318 1
121 0319 1 |---
122 0320 1 |
123 0321 1 |       Determine if an entity is present on the command line.
124 0322 1 |
125 0323 1 |       Inputs:
126 0324 1 |
127 0325 1 |       name = Address of entity name descriptor
128 0326 1 |
129 0327 1 |       Outputs:
130 0328 1 |
131 0329 1 |       routine value = True if present, else false.
132 0330 1 |---
133 0331 1
134 0332 2 BEGIN
135 0333 2
136 0334 2 LOCAL
137 0335 2     req_desc : BBLOCK [cli$c_reqdesc], ! Request descriptor block
138 0336 2     rpw :      BBLOCK [cli$c_workarea], ! Result parse work area
139 0337 2     req_flags : BITVECTOR [32];      ! Request flags array
140 0338 2
141 0339 2 CH$FILL ~, cli$c_reqdesc, req_desc);      ! Zero request descriptor block
142 0340 2 req_desc [int_b_type] = cli$k_present;    ! Set request type
143 0341 2 req_desc [int_l_getvm] = lib$get_vm;     ! Set address of get vm routine
144 0342 2 req_desc [int_l_freevm] = lib$free_vm;  ! Set address of free vm routine
145 0343 2
146 0344 2 | Set entity name
147 0345 2
148 0346 2 str$analyze sdesc (.name, req_desc [int_w_entlen], req_desc [int_l_entaddr]);
149 0347 2 RETURN (dcl$present (req_desc, rpw, req_flags)); ! Call DCL utility
150 0348 1 END;

```

```

.TITLE STACLINT
.IDENT \V04-000\

.EXTRN DCL$PRESENT, DCL$GETVALUE
.EXTRN DCL$DCLPARSE, DCL$DISPATCH
.EXTRN DCL$NEXTQUAL, DCL$ENDPARSE
.EXTRN STR$ANALYZE_SDESC
.EXTRN LIB$GET_VM, LIB$FREE_VM
.EXTRN STR$COPY_DX, CLIS_NOCOMD
.EXTRN CLIS_CONCAT, CLIS_COMMA
.EXTRN CLIS_PRESENT, CLIS_NEGATED
.EXTRN CLIS_LOCPRES, CLIS_LOCNEG
.EXTRN CLIS_DEFAULTED, CLIS_ABSENT

```

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

```

.ENTRY CLISPRESENT, Save R2,R3,R4,R5
MOVAB -160(SP), SP
MOVCS #0, (SP), #0, #28, REQ_DESC
MOVAB #80, REQ_DESC
MOVAB LIB$GET_VM, REQ_DESC+16
MOVAB LIB$FREE_VM, REQ_DESC+20

```

```

10 00 003C 0000
SE FF60 CE 9E 00002
6E 00 2C 00007
E4 AD E4 AD 0000C
F4 AD 0000000G 00 9E 00013
F8 AD 0000000G 00 9E 0001B

```

STACLINT
V04-000

	F0	AD	9F	00023	PUSHAB	REQ_DESC+12	:	0346
	EC	AD	9F	00026	PUSHAB	REQ_DESC+8	:	
	04	AC	DD	00029	PUSHL	NAME	:	
00000000G	00	03	FB	0002C	CALLS	#3, STR\$ANALYZE_SDESC	:	
		5E	DD	00033	PUSHL	SP	:	0347
	08	AE	9F	00035	PUSHAB	RPW	:	
	E4	AD	9F	00038	PUSHAB	REQ_DESC	:	
00000000G	00	03	FB	0003B	CALLS	#3, DCL\$PRESENT	:	
		04	00042	RET			:	0348

; Routine Size: 67 bytes, Routine Base: \$CODE\$ + 0000

```

152 0349 1 GLOBAL ROUTINE cli$get_value (name, retdesc, retlength) =
153 0350 1
154 0351 1 ---
155 0352 1
156 0353 1 This routine is called to obtain the next value
157 0354 1 associated with a named entity on the command line.
158 0355 1
159 0356 1 Inputs:
160 0357 1
161 0358 1 name = Address of entity name descriptor
162 0359 1 retdesc = Address of a return buffer descriptor
163 0360 1 retlength = Address of a longword to return the value
164 0361 1 length in.
165 0362 1
166 0363 1 Outputs:
167 0364 1
168 0365 1 True = A value was returned (there may be more).
169 0366 1 False = There is no more values associated with entity.
170 0367 1
171 0368 1 All other errors are signaled.
172 0369 1 ---
173 0370 1
174 0371 2 BEGIN
175 0372 2
176 0373 2 MAP
177 0374 2 retdesc : REF BBLOCK;
178 0375 2
179 0376 2 BUILTIN
180 0377 2 NULLPARAMETER;
181 0378 2
182 0379 2 LOCAL
183 0380 2 req_desc : BBLOCK [cli$c_reqdesc], ! Request descriptor block
184 0381 2 rpw : BBLOCK [cli$c_workarea], ! Result parse work area
185 0382 2 req_flags : BITVECTOR [32], ! Request flags array
186 0383 2 copy_status,
187 0384 2 status;
188 0385 2
189 0386 2 CH$FILL(0, cli$c_reqdesc, req_desc); ! Zero request descriptor block
190 0387 2 req_desc [int_b_type] = cli$c_getvalue; ! Set request type
191 0388 2 req_desc [int_l_getvm] = lib$get_vm; ! Set address of get vm routine
192 0389 2 req_desc [int_l_freevm] = lib$free_vm; ! Set address of free vm routine
193 0390 2
194 0391 2
195 0392 2 ! Set entity name
196 0393 2
197 0394 2 str$analyze_sdesc (.name, req_desc [int_w_entlen], req_desc [int_l_entaddr]);
198 0395 2 status = dc[$getvalue (req_desc, rpw, req_flags); ! Call DCL utility
199 0396 2
200 0397 2
201 0398 2 ! If return length was requested, then find it now.
202 0399 2
203 0400 2 IF NOT NULLPARAMETER (3) ! If return length requested
204 0401 2 THEN (.retlength) <0,16> = ! then original estimate
205 0402 2 .req_desc [int_w_retlen];
206 0403 2
207 0404 2 IF NOT (copy_status = str$copy_dx (.retdesc, ! Copy result into return desc
208 0405 2 req_desc [int_w_retlen]))

```



```

: 209      0406 2 THEN SIGNAL (.copy_status);           ! Signal any errors
: 210      0407 2
: 211      0408 2 IF NOT NULLPARAMETER (3)           ! If return length requested
: 212      0409 2 THEN BEGIN                          ! Then return correct value
: 213      0410 2 LOCAL temp : BBLOCK [dsc$c_s_bln];
: 214      0411 2 str$analyze_sdesc (.retdesc,         ! Get latest estimate
: 215      0412 2 temp [dsc$w_length], temp [dsc$a_pointer]);
: 216      0413 2 IF .temp LSSU .retdesc [dsc$w_length] ! Return the smaller estimate
: 217      0414 2 THEN (.retlength) <0,16> = .temp;
: 218      0415 2 END;
: 219      0416 2
: 220      0417 2 RETURN .status;
: 221      0418 1 END;

```

				007C 00000	.ENTRY	CLISGET VALUE, Save R2,R3,R4,R5,R6	0349
		56	00000000G	00 9E 00002	MOVAB	STR\$ANALYZE_SDESC, R6	
		5E	FF58	CE 9E 00009	MOVAB	-168(SP), SP	
1C		6E		00 2C 0000E	MOVCS	#0, (SP), #0, #28, REQ_DESC	0386
			E4	AD 00013			
		E4	AD	51 8F 90 00015	MOVB	#81, REQ_DESC	0387
		F4	AD	00000000G 00 9E 0001A	MOVAB	LIB\$GET_VM, REQ_DESC+16	0388
		F8	AD	00000000G 00 9E 00022	MOVAB	LIB\$FREE_VM, REQ_DESC+20	0389
			F0	AD 9F 0002A	PUSHAB	REQ_DESC+12	0394
			EC	AD 9F 0002D	PUSHAB	REQ_DESC+8	
			04	AC DD 00030	PUSHL	NAME	
		66		03 FB 00033	CALLS	#3, STR\$ANALYZE_SDESC	
				5E DD 00036	PUSHL	SP	0395
			10	AE 9F 00038	PUSHAB	RPW	
			E4	AD 9F 0003B	PUSHAB	REQ_DESC	
		00000000G	00	03 FB 0003E	CALLS	#3, DCL\$GETVALUE	
			52	50 D0 00045	MOVL	R0, STATUS	
			03	6C 91 00048	CMPB	(AP), #3	0400
				0A 1F 0004B	BLSSU	1\$	
			0C	AC D5 0004D	TSTL	12(AP)	
				05 13 00050	BEQL	1\$	
		0C	BC	EC AD B0 00052	MOVW	REQ_DESC+8, @RETLENGTH	0402
				EC AD 9F 00057	PUSHAB	REQ_DESC+8	0405
			08	AC DD 0005A	PUSHL	RETDESC	
		00000000G	00	02 FB 0005D	CALLS	#2, STR\$COPY_DX	
			09	50 E8 00064	BLBS	COPY_STATUS, -2\$	
				50 DD 00067	PUSHL	COPY_STATUS	0406
		00000000G	00	01 FB 00069	CALLS	#1, LIB\$SIGNAL	
			03	6C 91 00070	CMPB	(AP), #3	0408
				1F 1F 00073	BLSSU	3\$	
			0C	AC D5 00075	TSTL	12(AP)	
				1A 13 00078	BEQL	3\$	
			08	AE 9F 0007A	PUSHAB	TEMP+4	0412
			08	AE 9F 0007D	PUSHAB	TEMP	
			08	AC DD 00080	PUSHL	RETDESC	
			66	03 FB 00083	CALLS	#3, STR\$ANALYZE_SDESC	
04	AE	08	BC	00 ED 00086	CMPZV	#0, #16, @RETDESC, TEMP	0413
				05 1B 0008D	BLEQU	3\$	
		0C	BC	04 AE B0 0008F	MOVW	TEMP, @RETLENGTH	0414

STACLINT
V04-000

K 1
16-Sep-1984 00:32:58 VAX-11 Bliss-32 V4.0-742 Page 8
14-Sep-1984 12:15:40 DISK\$VMMASTER:[DCL.SRC]STACLINT.B32;1 (4)

50

52 D0 00094 3\$:
04 00097

MOVL STATUS, R0
RET

: 0417
: 0418

; Routine Size: 152 bytes, Routine Base: \$CODE\$ + 0043

```

223 0419 1 GLOBAL ROUTINE cli$dcl_parse (command, tables, promptrtn, continrtn, prompt) =
224 0420 1
225 0421 1 ---
226 0422 1
227 0423 1     This routine can be called to parse a command line.
228 0424 1
229 0425 1     Inputs:
230 0426 1
231 0427 1     command = address of descriptor of command string
232 0428 1     tables = address of command tables
233 0429 1     promptrtn = address of user missing parameter prompt routine
234 0430 1     continrtn = address of user line continuation routine
235 0431 1     prompt = address of user prompt string descriptor
236 0432 1
237 0433 1     Outputs:
238 0434 1
239 0435 1     The command is parsed and the command work area is initialized.
240 0436 1
241 0437 1 ---
242 0438 1
243 0439 2 BEGIN
244 0440 2
245 0441 2  E:ILTIN
246 0442 2  NULLPARAMETER;
247 0443 2
248 0444 2  LITERAL
249 0445 2  elements = 4;
250 0446 2
251 0447 2  LOCAL
252 0448 2  command_desc : BBLOCK [dsc$c_s bln],           ! Local command descriptor
253 0449 2  rtnlist : BBLOCK [4*(elements+1)],           ! Routine list
254 0450 2  req_desc : BBLOCK [cli$c_reqdesc],           ! Request descriptor block
255 0451 2  rpw : BBLOCK [cli$c_workarea],           ! Result parse work area
256 0452 2  req_flags : BITVECTOR [32];           ! Request flags array
257 0453 2
258 0454 2  C$FILL(0, cli$c_reqdesc, req_desc);           ! Zero request descriptor block
259 0455 2  req_desc [int_b_type] = cli$c_dclparse;           ! Set request type
260 0456 2  req_desc [int_l_getvm] = lib$get_vm;           ! Set address of get vm routine
261 0457 2  req_desc [int_l_freevm] = lib$free_vm;           ! Set address of free vm routine
262 0458 2
263 0459 2  IF NOT NULLPARAMETER(1)           ! Set command line descriptor
264 0460 2  THEN BEGIN
265 0461 2  str$analyze_sdesc (.command, command_desc [dsc$w_length],
266 0462 2  command_desc [dsc$a_pointer]);
267 0463 2  req_desc [int_l_entaddr] = command_desc;
268 0464 2  END;
269 0465 2
270 0466 2  req_desc [int_l_tables] = .tables;           ! Set address of command tables
271 0467 2
272 0468 2  IF NOT NULLPARAMETER(3) OR NOT NULLPARAMETER(4) ! If prompt or continue routine
273 0469 2  THEN BEGIN
274 0470 2  C$FILL(0, 4*(elements+1), rtnlist);           ! Zero the list
275 0471 2  rtnlist [int_l_listlen] = elements;           ! Set number of elements in list
276 0472 2  req_desc [int_l_list] = rtnlist;           ! And connect it
277 0473 2  END;
278 0474 2
279 0475 2  IF NOT NULLPARAMETER(3)

```

```

: 280 0476 2 THEN rtnlist [int_l_promptrtn] = .promptrtn; ! Set address of prompt routine
: 281 0477 2 IF NOT NULLPARAMETER(4)
: 282 0478 2 THEN rtnlist [int_l_continrtn] = .continrtn; ! Set address of continue routine
: 283 0479 2 IF NOT NULLPARAMETER(5) ! Set prompt descriptor
: 284 0480 2 THEN str$analyze_sdesc (.prompt, rtnlist [int_w_pmptlen],
: 285 0481 2 rtnlist [int_l_pmpfaddr]);
: 286 0482 2
: 287 0483 2 RETURN (dcl$dclparse (req_desc, rpw, req_flags));! Call DCL utility
: 288 0484 1 END;

```

			007C	00000	.ENTRY	CLISDCL PARSE, Save R2,R3,R4,R5,R6	: 0419
		56	00	9E	MOVAB	STR\$ANALYZE_SDESC, R6	
		5E	CE	9E	MOVAB	-188(SP), SP	
1C	00	6E	00	2C	MOVCS	#0, (SP), #0, #28, REQ_DESC	: 0454
			C8	AD			
			AD	8F	MOVB	#83, REQ_DESC	: 0455
		C8	AD	9E	MOVAB	LIB\$GET_VM, REQ_DESC+16	: 0456
		D8	AD	9E	MOVAB	LIB\$FREE_VM, REQ_DESC+20	: 0457
		DC	AD	95	TSTB	(AP)	: 0459
			6C	95			
			16	13	BEQL	1\$	
			04	AC	TSTL	4(AP)	
			11	13	BEQL	1\$	
			FC	AD	PUSHAB	COMMAND_DESC+4	: 0462
			F8	AD	PUSHAB	COMMAND_DESC	: 0461
			04	AC	PUSHL	COMMAND-	
			03	FB	CALLS	#3, STR\$ANALYZE_SDESC	
		66	AD	9E	MOVAB	COMMAND_DESC, REQ_DESC+12	: 0463
		D4	AD	D0	MOVL	TABLES, -REQ_DESC+4	: 0466
		CC	AD	91	CMPB	(AP), #3	: 0468
			03	1F	BLSSU	2\$	
			0C	AC	TSTL	12(AP)	
			0A	12	BNEQ	3\$	
			04	6C	CMPB	(AP), #4	
			15	1F	BLSSU	4\$	
			10	AC	TSTL	16(AP)	
			10	13	BEQL	4\$	
14	00	6E	00	2C	MOVCS	#0, (SP), #0, #20, RTNLIST	: 0470
			E4	AD			
			04	D0	MOVL	#4, RTNLIST	: 0471
		E4	AD	9E	MOVAB	RTNLIST, REQ_DESC+24	: 0472
		EG	AD	91	CMPB	(AP), #3	: 0475
			03	1F	BLSSU	5\$	
			0C	AC	TSTL	12(AP)	
			05	13	BEQL	5\$	
			0C	AC	MOVL	PROMPTRTN, RTNLIST+4	: 0476
		E8	AD	91	CMPB	(AP), #4	: 0477
			04	1F	BLSSU	6\$	
			10	AC	TSTL	16(AP)	
			05	13	BEQL	6\$	
			10	AC	MOVL	CONTINRTN, RTNLIST+8	: 0478
		EC	AD	91	CMPB	(AP), #5	: 0479
			05	1F	BLSSU	7\$	
			14	AC	TSTL	20(AP)	

STACLINT
V04-000

N 1
16-Sep-1984 00:32:58
14-Sep-1984 12:15:40

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

		OC	13	00093	BEQL	7\$:
	F4	AD	9F	00095	PUSHAB	RTNLIST+16	: 0481
	F0	AD	9F	00098	PUSHAB	RTNLIST+12	: 0480
	14	AC	DD	0009B	PUSHL	PROMPT	:
66		03	FB	0009E	CALLS	#3, STR\$ANALYZE_SDESC	:
		5E	DD	000A1	PUSHL	SP	: 0483
	08	AE	9F	000A3	PUSHAB	RPW	:
	C8	AD	9F	000A6	PUSHAB	REQ_DESC	:
00000000G	00	03	FB	000A9	CALLS	#3, -DCL\$DCLPARSE	:
		04	000B0	RET			: 0484

; Routine Size: 177 bytes, Routine Base: \$CODE\$ + 00DB

```

290 0485 1 GLOBAL ROUTINE cli$dispatch (argument) =
291 0486 1
292 0487 1 ---
293 0488 1
294 0489 1 This routine can be called to dispatch to any verb processing
295 0490 1 routines if the command has the ROUTINE attribute.
296 0491 1
297 0492 1 Inputs:
298 0493 1
299 0494 1 argument = address of user supplied argument to his own routine
300 0495 1
301 0496 1 Outputs:
302 0497 1
303 0498 1 The verb routine is called (if any).
304 0499 1
305 0500 1 The status passed back from the routine is returned in R0.
306 0501 1 If no routine is specified, success is returned.
307 0502 1 ---
308 0503 1
309 0504 2 BEGIN
310 0505 2
311 0506 2 BUILTIN
312 0507 2 NULLPARAMETER;
313 0508 2
314 0509 2 LOCAL
315 0510 2 req_desc : BBLOCK [cli$c_reqdesc], ! Request descriptor block
316 0511 2 rpw : BBLOCK [cli$c_workarea], ! Result parse work area
317 0512 2 req_flags : BITVECTOR [32]; ! Request flags array
318 0513 2
319 0514 2 CH$FILL(0, cli$c_reqdesc, req_desc); ! Zero request descriptor block
320 0515 2 req_desc [int_b_type] = cli$k_dispatch; ! Set request type
321 0516 2 req_desc [int_l_getvm] = lib$get_vm; ! Set address of get vm routine
322 0517 2 req_desc [int_l_freevm] = lib$free_vm; ! Set address of free vm routine
323 0518 2 IF NOT NULLPARAMETER (1)
324 0519 2 THEN req_desc [int_l_entaddr] = .argument; ! Set address of user argument
325 0520 2 RETURN (dcl$dispatch (req_desc, rpw, req_flags)); ! Call callback utility
326 0521 1 END;

```

1C	00	SE	FF60	CE	9E	00002	.ENTRY	CLISDISPATCH, Save R2,P3,R4,R5	: 0485
		6E		00	2C	00007	MOVAB	-160(SP), SP	: 0514
			E4	AD		0000C	MOVCS	#0, (SP). #0, #28, REQ_DESC	: 0515
		E4	AD	8F	90	0000E	MOVB	#84, REQ_DESC	: 0516
		F4	AD	00	9E	00013	MOVAB	LIB\$GET_VM, REQ_DESC+16	: 0517
		F8	AD	00	9F	0001B	MOVAB	LIB\$FREE_VM, REQ_DESC+20	: 0518
				6C	95	00023	TSTB	(AP)	: 0519
				0A	13	00025	BEQL	1\$: 0520
			04	AC	D5	00027	TSTL	4(AP)	: 0519
				05	13	0002A	BEQL	1\$: 0520
		F0	AD	04	AC	D0	MOVL	ARGUMENT, REQ_DESC+12	: 0519
				5E	DD	00031	PUSHL	SP	: 0520
				08	AE	9F	PUSHAB	RPW	: 0520
			E4	AD	9F	00036	PUSHAB	REQ_DESC	: 0520

STACLINT
VC4-000

C 2
16-Sep-1984 00:32:58
14-Sep-1984 12:15:40

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCL.SRC]STACLINT.E32;1 Page 13 (6)

00000000G 00

03 FB 00039
04 00040

CALLS #3, DCL\$DISPATCH
RET

: 0521

; Routine Size: 65 bytes. Routine Base: \$CODES + 018C

```

: 328 0522 1 GLOBAL ROUTINE cli$end_parse =
: 329 0523 1
: 330 0524 1 ---
: 331 0525 1
: 332 0526 1 This routine is called when the user has completed
: 333 0527 1 all command line parsing. It checks that all qualifiers
: 334 0528 1 which appeared on the command line were processed in one
: 335 0529 1 way or another by the utility.
: 336 0530 1
: 337 0531 1 Inputs:
: 338 0532 1
: 339 0533 1 None
: 340 0534 1
: 341 0535 1 Outputs:
: 342 0536 1
: 343 0537 1 None
: 344 0538 1 ---
: 345 0539 1
: 346 0540 2 BEGIN
: 347 0541 2
: 348 0542 2 LOCAL
: 349 0543 2 req_desc : BBLOCK [cli$c_reqdesc], ! Request descriptor block
: 350 0544 2 rpw : BBLOCK [cli$c_workarea], ! Result parse work area
: 351 0545 2 req_flags : BITVECTOR [32]; ! Request flags array
: 352 0546 2
: 353 0547 2 CH$FILL(0, cli$c_reqdesc, req_desc); ! Zero request descriptor block
: 354 0548 2 req_desc [cli$a_prsact] = lib$get_vm; ! Set address of get vm routine
: 355 0549 2 req_desc [cli$a_absact] = lib$free_vm; ! Set address of free vm routine
: 356 0550 2 RETURN (dcl$endparse (req_desc, rpw, req_flags)); ! Call DCL utility
: 357 0551 1 END;

```

				003C 00000	.ENTRY CLISEND_PARSE, Save R2,R3,R4,R5	: 0522
		SE	FF60	CE 9E 0C002	MOVAB -160(SP), SP	: 0547
1C	00	6E		00 2C 00007	MOVCS #0, (SP), #0, #28, REQ_DESC	: 0548
			E4	AD 00C0C	MOVAB LIB\$GET_VM, REQ_DESC+16	: 0549
		F4	AD 00000000G	00 9E 0000E	MOVAB LIB\$FREE_VM, REQ_DESC+20	: 0550
		F8	AD 00000000G	00 9E 00016	PUSHL SP	: 0551
				5E DD 0001E	PUSHAB RPW	
			08	AE 9F 00020	PUSHAB REQ_DESC	
			F4	AD 9F 00023	CALLS #3, DCL\$ENDPARSE	
		00000000G	00	03 FB 00026	RET	: 0551
				04 0002D		

; Routine Size: 46 bytes, Routine Base: \$CODE\$ + 010D


```

359 0552 1 GLOBAL ROUTINE cli$next_qual (name) =
360 0553 1
361 0554 1 |---
362 0555 1 |
363 0556 1 |       Move to the next command qualifier on the line.
364 0557 1 |
365 0558 1 | Inputs:
366 0559 1 |
367 0560 1 |       name = Address of entity name descriptor
368 0561 1 |
369 0562 1 | Outputs:
370 0563 1 |
371 0564 1 |       routine value = True if present, else false.
372 0565 1 |---
373 0566 1
374 0567 2 BEGIN
375 0568 2
376 0569 2 MAP
377 0570 2     name : REF BBLOCK;
378 0571 2
379 0572 2 LOCAL
380 0573 2     req_desc : BBLOCK [cli$c_reqdesc], ! Request descriptor block
381 0574 2     rpw : BBLOCK [cli$c_workarea], ! Result parse work area
382 0575 2     req_flags : BITVECTOR [32]; ! Request flags array
383 0576 2
384 0577 2 CH$FILL(0, cli$c_reqdesc, req_desc); ! Zero request descriptor block
385 0578 2 req_desc [int_b_type] = cli$k_nextqual; ! Set request type
386 0579 2 req_desc [int_l_getvm] = lib$get_vm; ! Set address of get vm routine
387 0580 2 req_desc [int_l_freevm] = lib$free_vm; ! Set address of free vm routine
388 0581 2
389 0582 2 | Set entity name
390 0583 2
391 0584 2 str$analyze sdesc (.name, req_desc [int_w_entlen], req_desc [int_l_entaddr]);
392 0585 2 RETURN (dcl$nextqual (req_desc, rpw, req_flags)); ! Call DCL utility
393 0586 1 END;

```

				003C 00000	.ENTRY CLISNEXT_QUAL, Save R2,R3,R4,R5	: 0552
		SE FF60	CE 9E 00002		MOVAR -160(SP), SP	: 0577
1C	00	6E	00 2C 00007		MOVCS #0, (SP), #0, #28, REQ_DESC	: 0578
		E4	AD 00000		MOVB #85, REQ_DESC	: 0579
		E4 AD	55 8F 90 0000E		MOVAB LIB\$GET_VM, REQ_DESC+16	: 0580
		F4 AD	000000C0G 00 9E 00013		MOVAB LIB\$FREE_VM, REQ_DESC+20	: 0584
		F8 AD	00000000G 00 9E 0001B		PUSHAB REQ_DESC+12	: 0585
			F0 AD 9F 00023		PUSHAB REQ_DESC+8	
			EC AD 9F 00026		PUSHL NAME	
			04 AC DD 00029		CALLS #3, STR\$ANALYZE_SDESC	
	00000000G	00	03 FB 0002C		PUSHL SP	: 0586
			SE DD 00033		PUSHAB RPW	
			08 AE 9F 00035		PUSHAB REQ_DESC	
		E4	AD 9F 00038		CALLS #3, DCL\$NEXTQUAL	
	00000000G	00	03 FB 0003B		RET	: 0586
			04 00042			

STACLINT
V04-000

F 2
16-Sep-1984 00:32:58
14-Sep-1984 12:15:40

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCL.SRC]STACLINT.B32;1 Page 16 (8)

: Routine Size: 67 bytes, Routine Base: \$CODE\$ + 01FB

: 395 0587 1 END
: 396 0588 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

: Name Bytes Attributes
: \$CODE\$ 574 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)

Library Statistics

File	Symbols		Pages Mapped	Processing Time
	Total	Loaded Percent		
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	17 0	581	00:01.0

COMMAND QUALIFIERS

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

: Size: 574 code + 0 data bytes
: Run Time: 00:11.0
: Elapsed Time: 00:38.7
: Lines/CPU Min: 3207
: Lexemes/CPU-Min: 19881
: Memory Used: 95 pages
: Compilation Complete

