

DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDDDD		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
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL

```

LL               IIIIII
LL               IIIIII
LL               III
LL               III
LL               III
LL               III
LL               III
LL               III
LL               III
LL               III
LL               III
LL               III
LL               III
LL               III
LLLLLLLLLLLLLL IIIIII
LLLLLLLLLLLLLL IIIIII
SSSSSSSS
SSSSSSSS
SS
SS
SS
SS
SSSSSS
SSSSSS
SS
SS
SS
SS
SSSSSSSS
SSSSSSSS

```

```

1 0001 0 MODULE rpclint (IDENT='V04-000',
2 0002 0 ADDRESSING_MODE(NONEXTERNAL=LONG_RELATIVE,
3 0003 0 EXTERNAL=GENERAL) =
4 0004 0
5 0005 1 BEGIN
6 0006 1
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1
32 0032 1 **
33 0033 1 FACILITY: Command language interface routines
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 These routines are used to enable a newly activated
38 0038 1 image to obtain the command parameters and qualifiers
39 0039 1 from the command language interpreter.
40 0040 1
41 0041 1 ENVIRONMENT:
42 0042 1
43 0043 1 VAX/VMS operating system. unprivileged user mode.
44 0044 1
45 0045 1 AUTHOR: Tim Halvorsen, Mar 1980
46 0046 1
47 0047 1 Modified by:
48 0048 1
49 0049 1 V03-015 HWS0073 Harold Schultz 12-Jun-1984
50 0050 1 When error encountered, always return an error code
51 0051 1 rather than a 0. When a syntax error is signaled, output
52 0052 1 secondary error message of entity not found (ENTNF).
53 0053 1 Put length check back into FIND_ENTITY optimization (undo
54 0054 1 HWS0070)
55 0055 1
56 0056 1 V03-014 HWS0070 Harold Schultz 29-May-1984
57 0057 1 Don't check for length in FIND_ENTITY optimization.

```

58	0058	1	
59	0059	1	
60	0060	1	V03-013 HWS0028 Harold Schultz 12-Mar-1984
61	0061	1	Optimize FIND_ENTITY and UPCASE.
62	0062	1	
63	0063	1	V03-012 PCG0022 Peter George 09-Feb-1984
64	0064	1	Fix bug in default keyword processing.
65	0065	1	
66	0066	1	V03-011 PCG0021 Peter George 27-Jul-1983
67	0067	1	Look past first instance of a keyword.
68	0068	1	
69	0069	1	V03-010 PCG0020 Peter George 29-Jun-1983
70	0070	1	Use event flags more intelligently.
71	0071	1	Use multi-national upcase algorithm.
72	0072	1	
73	0073	1	V03-009 PCG0019 Peter George 20-Apr-1983
74	0074	1	Add explicit check for dispatch routine address of zero.
75	0075	1	
76	0076	1	V03-008 PCG0018 Peter George 17-Feb-1983
77	0077	1	Convert to new table structure.
78	0078	1	Use PTR_B_NUMBER to get qualifier or keyword number.
79	0079	1	
80	0080	1	V03-007 PCG0017 Peter George 27-Dec-1982
81	0081	1	Be smarter about using old get value contexts.
82	0082	1	
83	0083	1	V03-006 PCG0016 Peter George 13-Dec-1982
84	0084	1	Fix bug in multiple nested value fetch.
85	0085	1	Clean up some more code.
86	0086	1	Return CLIS_ABSENT instead of false when no value is found.
87	0087	1	
88	0088	1	V03-005 PCG0015 Peter George 11-Nov-1982
89	0089	1	Be smarter about when to return CLIS_COMMA for
90	0090	1	parameter values. Do not return a default value
91	0091	1	if a qualifier or keyword has been explicitly
92	0092	1	negated.
93	0093	1	
94	0094	1	V03-004 PCG0014 Peter George 14-Oct-1982
95	0095	1	Return CLIS_COMMA for default values.
96	0096	1	Add DCL\$NEXTQUAL.
97	0097	1	
98	0098	1	V03-003 PCG0013 Peter George 01-Sep-1982
99	0099	1	Support keyword parsing.
100	0100	1	
101	0101	1	V03-002 PCG0012 Peter George 03-Aug-1982
102	0102	1	Redo the previous fix in a different manner.
103	0103	1	
104	0104	1	V03-001 PCG0011 Peter George 14-Jun-1982
105	0105	1	Differentiate between local and global presence
106	0106	1	in CLISPRESENT.
107	0107	1	--

RPCLINT
V04-000

H 12
16-Sep-1984 00:26:36
14-Sep-1984 12:15:33

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCL.SRC]RPCLINT.B32;1 Page 3 (2)

```
: 109      0108 1 :  
: 110      0109 1 : Include files  
: 111      0110 1 :  
: 112      0111 1 LIBRARY 'SYSSLIBRARY:LIB':  
: 113      0112 1 REQUIRE 'SHRLIBS:UTILDEF':  
: 114      0297 1 REQUIRE 'LIBS:CLITABDEF':  
: 115      0622 1 REQUIRE 'LIBS:INTDEF':  
: 116      0648 1 REQUIRE 'LIBS:DCLDEF':
```

```
: VMS common definitions  
: Common VMS BLISS definitions  
: CLI definitions  
: CLI definitions  
: DCL definitions
```

```

118 1720 1  !
119 1721 1  ! Table of contents
120 1722 1  !
121 1723 1  LINKAGE
122 1724 1  entity_linkage = call : GLOBAL(block=9,number=10,type=11);
123 1725 1  ! Common linkage
124 1726 1  EXTERNAL ROUTINE
125 1727 1  sys$cli;
126 1728 1  ! Callback entry point
127 1729 1  FORWARD ROUTINE
128 1730 1  initialize : NOVALUE,
129 1731 1  dcl$present,
130 1732 1  parameter_present : entity_linkage,
131 1733 1  qualifier_present : entity_linkage,
132 1734 1  dcl$getValue,
133 1735 1  parameter_value,
134 1736 1  qualifier_value,
135 1737 1  reserved_value,
136 1738 1  verify_entities : entity_linkage,
137 1739 1  find_main_entity : entity_linkage,
138 1740 1  verify_keywords,
139 1741 1  find_keyword_entity : entity_linkage,
140 1742 1  find_entity : entity_linkage,
141 1743 1  guess_entity : entity_linkage,
142 1744 1  guess_keyword_entity,
143 1745 1  process_keyword_list,
144 1746 1  get_param_token,
145 1747 1  get_next_value,
146 1748 1  get_explicit_value,
147 1749 1  get_specified_value,
148 1750 1  get_default_value,
149 1751 1  insert_next_level,
150 1752 1  insert_string,
151 1753 1  insert_char,
152 1754 1  allocate_default_buffer,
153 1755 1  local_qualifier,
154 1756 1  global_qualifier,
155 1757 1  token_string : NOVALUE,
156 1758 1  upcase : NOVALUE,
157 1759 1  batch_job,
158 1760 1  convert_keyword_list,
159 1761 1  dcl$dispatch,
160 1762 1  dcl$nextqual,
161 1763 1  dcl$endparse,
162 1764 1  dcl$getline;
163 1765 1  !
164 1766 1  !
165 1767 1  ! Change name of the PSECT's to conform to DCL standards.
166 1768 1  !
167 1769 1  PSECT PLIT = DCL$ZCODE(EXECUTE, ALIGN(0));
168 1770 1  PSECT CODE = DCL$ZCODE(EXECUTE, ALIGN(0));
169 1771 1  !
170 1772 1  !
171 1773 1  ! Get values of status messages.
172 1774 1  !
173 1775 1  EXTERNAL LITERAL
174 1776 1  cli$_comma,
! Value is terminated with a comma

```



```

175 1777 1  cli$_concat,      ! Value is terminated with a plus
176 1778 1  cli$_present, ! Entity is explicitly present
177 1779 1  cli$_negated,  ! Entity is explicitly not present
178 1780 1  cli$_locpres,  ! Qualifier is locally present
179 1781 1  cli$_locneg,   ! Qualifier is explicitly not locally present
180 1782 1  cli$_defaulted, ! Entity is implicitly present
181 1783 1  cli$_absent,    ! Entity is implicitly not present
182 1784 1  cli$_invrount,  ! Invalid routine
183 1785 1  cli$_entnf,     ! Entity not found
184 1786 1  exe$c_sysefn;   ! System event flag number
185 1787 1
186 P 1788 1  $shr_messages(msg,3,      ! Prefix MSG$_ with CLI facility
187 1789 1  (syntax,severe));
188 1790 1
189 1791 1  LITERAL
190 1792 1  msg$_noentity = msg$_syntax; ! Provide temporary definition
191 1793 1
192 1794 1
193 1795 1  ! Define entity type numbers (for internal classification of entities)
194 1796 1
195 1797 1  LITERAL
196 1798 1  min_entity = 1,             ! Minimum entity type number
197 1799 1  param_entity = 1,          ! Entity is a parameter
198 1800 1  qual_entity = 2,           ! Entity is a qualifier
199 1801 1  reserved_entity = 3,       ! Entity is a reserved word
200 1802 1  max_entity = 3;            ! Maximum entity type number
201 1803 1
202 1804 1
203 1805 1  ! Macros to get the address of a token descriptor given a token index,
204 1806 1  and to get a token index given the address of a token descriptor.
205 1807 1
206 M 1808 1  MACRO token_desc(index) = ! Index -> Token
207 1809 1  wrk [wrk_g_result] + (index-1)*ptr_c_length%;
208 M 1810 1  MACRO table_index(token) = ! Token -> Index
209 1811 1  (token = wrk [wrk_g_result])/ptr_c_length + 1%;
210 1812 1
211 1813 1
212 1814 1  ! Macro to zero the unused portions of the context arrays.
213 1815 1
214 M 1816 1  MACRO zero_context_arrays(index) =
215 M 1817 1  BEGIN
216 M 1818 1  CH$FILL (0, 4*(dcl_c_context-(index)), entity_context [index]);
217 M 1819 1  CH$FILL (0, 4*(dcl_c_context-(index)), token_context [index]);
218 1820 1  END%;
219 1821 1
220 1822 1
221 1823 1  ! Cells containing addresses of CLINT own storage and command work area.
222 1824 1  ! If these addresses were not defined by DCL$DCL_PARSE, then we are parsing
223 1825 1  a supervisor mode command and they are initialized here.
224 1826 1
225 1827 1  EXTERNAL
226 1828 1  cti$gl_clintown : REF BBLOCK, ! Address of pointer to own storage
227 1829 1  cti$gl_dclprstown : REF BBLOCK; ! Address of pointer to wrk area
228 1830 1
229 1831 1
230 1832 1  ! Table of reserved entity names
231 1833 1

```

NPCLINT
V04-000

K 12
16-Sep-1984 00:26:36
14-Sep-1984 12:15:33

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCL.SRC]RPCLINT.B32;1 Page 6 (3)

```
: 232      1834 1 BIND
: 233      1835 1 reserved words = UPLIT BYTE(
: 234      1836 1      %ASCIC '$LINE',
: 235      1837 1      %ASCIC '$VERB',
: 236      1838 1      0);
: 237      1839 1
```

```
! Returns entire command line
! Returns verb name (as defined)
! -- End of list
```



```

239 1840 1 ROUTINE initialize (get_vm, free_vm) : NOVALUE =
240 1841 1
241 1842 1 ---
242 1843 1
243 1844 1 This routine is called on the first call to this
244 1845 1 interface package. It initializes the own storage
245 1846 1 and sets up for result parsing.
246 1847 1
247 1848 1 Inputs:
248 1849 1
249 1850 1 get_vm = Address of LIB$GET VM routine
250 1851 1 free_vm = Address of LIB$FREE VM routine
251 1852 1
252 1853 1 Outputs:
253 1854 1
254 1855 1 OWN storage initialized.
255 1856 1 ---
256 1857 1
257 1858 2 BEGIN
258 1859 2
259 1860 2 BUILTIN
260 1861 2 PROBEW, ! True if location writable
261 1862 2 PROBER; ! True if location readable
262 1863 2
263 1864 2 LOCAL
264 1865 2 index, ! Token index
265 1866 2 plm : REF BBLOCK, ! Address of parameter limit block
266 1867 2 req_desc : BBLOCK [cli$c_reqdesc], ! Callback request descriptor
267 1868 2 req_flags : BITVECTOR [32], ! Callback request flags
268 1869 2 rpw : BBLOCK [cli$c_workarea], ! Result parse work area
269 1870 2 token : REF BBLOCK, ! Address of token descriptor
270 1871 2 wrk : REF BBLOCK, ! Address of WRK block
271 1872 2 status;
272 1873 2
273 1874 2
274 1875 2
275 1876 2 Get memory for parse routines' own storage. Store address of own storage
276 1877 2 in CTL$GL_CLINTOWN. If LIB$GET_VM is unsuccessful, then abort.
277 1878 2
278 1879 3 IF NOT (status = (.get_vm) (%REF(dcl_c_size), ctl$gl_clintown)) ! Get memory for CLINT own storage
279 1880 2 THEN SIGNAL (.status); ! Signal error if failed
280 1881 2
281 1882 2
282 1883 2 If the WRK block pointer is zero, then make an old fashioned INITPRS
283 1884 2 callback to get it.
284 1885 2
285 1886 2 IF .ctl$gl_dclprsown EQL 0 ! If we have no WRK block pointer
286 1887 3 THEN BEGIN ! Then get one now
287 1888 3 CH$FILL (0, cli$c_reqdesc, req_desc); ! Zero request desc block
288 1889 3 req_desc [cli$b_reqtype] = cli$k_initprs; ! Set request type
289 1890 4 IF NOT (status = SYS$CLI (req_desc, rpw, req_flags)) ! Init result parsing solely to get
290 1891 3 THEN SIGNAL (.status); ! rpw [rpw_l_dclwrk]
291 1892 3 ctl$gl_dclprsown = .rpw [rpw_l_dclwrk]; ! Store address of WRK area
292 1893 2 END;
293 1894 2
294 1895 2
295 1896 2 ! Get the address of the command WRK block from CTL$GL_DCLPRSOWN.

```

```
296 1897 2 !
297 1898 2 wrk = .ctl$gl_dclprstown; ! Get address of WRK area
298 1899 2
299 1900 2
300 1901 2 ! Verify the validity of the CLI WRK area pointer, to ensure that we aren't
301 1902 2 ! trying to deal with a mismatched WRK structure.
302 1903 2
303 1904 2 ctl$gl_clintown [dcl_v_nowrkarea] = true; ! Assume invalid WRK area
304 1905 2
305 1906 2
306 1907 2 ! Check first result parse descriptor.
307 1908 2
308 1909 2 token = wrk [wrk_g_result]; ! Point to first entry in array
309 1910 2 IF NOT PROBER(%REF(psl$sc_user),%REF(ptr_c_length),.token) ! If not readable,
310 1911 2 OR .token [ptr_v_type] GTRU ptr_k_ignore ! Or invalid type code,
311 1912 2 OR .token [ptr_v_term] GTRU ptr_k_lparen ! Or invalid terminator code,
312 1913 2 OR .token [ptr_v_term] LSSU ptr_k_blank
313 1914 2 OR .wrk [wrk_l_rslnxt] LSSA wrk [wrk_g_result] ! Or invalid RSL pointer,
314 1915 2 OR .wrk [wrk_l_rslnxt] GTRA wrk [wrk_g_result] + wrk_c_rslbufsiz
315 1916 2 THEN RETURN; ! Return with invalid WRK
316 1917 2
317 1918 2
318 1919 2 ! Check first parameter entity block and first qualifier entity block.
319 1920 2
320 1921 2 token = .wrk [wrk_l_proptr]; ! Get address of param entities
321 1922 2 IF .token NEQ 0 ! If invalid pointer,
322 1923 2 THEN IF NOT PROBER(%REF(psl$sc_user),%REF(10),.token)
323 1924 2 THEN RETURN; ! Return with invalid WRK
324 1925 2
325 1926 2 token = .wrk [wrk_l_quablk]; ! Get address of qual entities
326 1927 2 IF .token NEQ 0 ! If invalid pointer,
327 1928 2 THEN IF NOT PROBER(%REF(psl$sc_user),%REF(10),.token)
328 1929 2 THEN RETURN; ! Return with invalid WRK
329 1930 2
330 1931 2
331 1932 2 ! If we've gotten this far, then indicate that the WRK area is valid.
332 1933 2
333 1934 2 ctl$gl_clintown [dcl_v_nowrkarea] = false; ! Indicate valid WRK area
334 1935 2
335 1936 2
336 1937 2 ! Initialize the CLINT own storage area.
337 1938 2
338 1939 2 ! Clear all information about the default value buffer.
339 1940 2
340 1941 2 ctl$gl_clintown [dcl_w_bufllen] = 0; ! Clear length of buffer
341 1942 2 CH$FILL (0, dsc$sc_s_bln, ctl$gl_clintown [dcl_w_defllen]); ! Zero default value descriptor
342 1943 2
343 1944 2
344 1945 2 ! Save the addresses of the LIB$GET_VM and LIB$FREE_VM routines.
345 1946 2
346 1947 2 ctl$gl_clintown [dcl_l_getvm] = .get_vm; ! Store LIB$GET_VM
347 1948 2 ctl$gl_clintown [dcl_l_freevm] = .free_vm; ! Store LIB$FREE_VM
348 1949 2
349 1950 2
350 1951 2 ! Clear all context information.
351 1952 2
352 1953 2 ctl$gl_clintown [dcl_v_nextqual] = false; ! Assume normal qualifier parse
```

```
1954 2 CH$FILL(0, 4*dcl_c_context, ctl$gl_clintown [dcl_l_entity]);      ! Set no entities processed yet
1955 2 CH$FILL(0, 4*dcl_c_context, ctl$gl_clintown [dcl_l_token]);      ! Set no tokens processed yet
1956 2 CH$FILL(0, 16*plm_c_size, ctl$gl_clintown [dcl_l_prmlim]);      ! Zero parameter limit descs (plms)
1957 2 ctl$gl_clintown [dcl_b_param] = 0;      ! Set no parameters processed yet
1958 2 ctl$gl_clintown [dcl_l_qual] = 0;      ! Set no qualifier processed yet
1959 2
1960 2
1961 2 Initialize the parameter list markers in the clint own storage.
1962 2
1963 2 For each parameter type, a plm longword is filled in. Each byte
1964 2 contains the index of a result parse descriptor, as follows.
1965 2
1966 2     plm_b_nxtdesc = next parameter value to examine
1967 2     plm_b_fstdesc = first parameter in the list
1968 2     plm_b_lstdesc = last parameter value before next parameter type
1969 2     plm_b_quadesc = first possible local qualifier token
1970 2
1971 2 index = 0;
1972 2 plm = ctl$gl_clintown [dcl_l_prmlim];      ! Start at first token descriptor
1973 2                                     ! Point to first plm longword
1974 2 status = get_param_token(index, token);      ! Get first parameter token
1975 2
1976 2 WHILE (.status)      ! Until no more parameters
1977 2 DO BEGIN
1978 2     plm [plm_b_fstdesc] = .index;      ! Save starting token for parameter
1979 2     plm [plm_b_nxtdesc] = .index;      ! and set next value to process
1980 2     plm [plm_b_quadesc] = .index;      ! and set first possible qualifier token
1981 2
1982 2     WHILE (status = get_param_token(index, token))      ! Scan for next parameter value
1983 2     DO BEGIN
1984 2         BIND preceeding_token = .token - ptr_c_length: BBLOCK;
1985 2         IF .preceeding_token [ptr_v_term] EQ[ ptr_k_blank
1986 2         THEN EXITLOOP;
1987 2     END;
1988 2
1989 2     plm [plm_b_lstdesc] = .index-1;      ! Save ending token for prev. parameter
1990 2     plm = .plm + plm_c_size;      ! Skip to next plm
1991 2     END;
1992 2
1993 2 RETURN true;
1994 1 END;
```

```
.TITLE RPCLINT
.IDENT \V04-000\
.PSECT DCL$ZCODE, NOWRT, 0

45 4E 49 4C 24 05 0000 P.AAA: .ASCII <5>\$LINE\
42 52 45 56 24 05 0006 .ASCII <5>\$VERB\
00 000C .BYTE 0
```

```
RESERVED_WORDS= P.AAA
.EXTRN SYSSCLI, CLIS_COMMA
.EXTRN CLIS_CONCAT, CLIS_PRESENT
.EXTRN CLIS_NEGATED, CLIS_LOCPRES
.EXTRN CLIS_LOCNEG, CLIS_DEFAULTED
```

.EXTRN CLIS_ABSENT, CLIS_INVRUT
.EXTRN CLIS_ENTNF, EXESC_SYSEFN
.EXTRN CTLSGL_CLINTOWN
.EXTRN CTLSGL_DCLPRSOWN

OFFC U0000 INITIALIZE:

			5B	00000000V	EF	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	1840	
			5A	00000000G	00	9E	00009	MOVAB	GET PARAM TOKEN, R1		
			59	00000000G	00	9E	00010	MOVAB	LIB\$SIGNAL, R10		
			5E	FF54	CE	9E	00017	MOVAB	CTLSGL_DCLPRSOWN, R9		
				00000000G	00	9F	0001C	MOVAB	-172(SP), SP		
04			AE	90	8F	9A	00022	PUSHAB	CTLSGL_CLINTOWN	1879	
				04	AE	9F	00027	MOVZBL	#144, 4(SP)		
04			BC		02	FB	0002A	PUSHAB	4(SP)		
			58		50	D0	0002E	CALLS	#2, @GET VM		
			05		58	E8	00031	MOVL	R0, STATUS		
					58	DD	00034	BLBS	STATUS, 1\$		
			6A		01	FB	00036	PUSHL	STATUS	1880	
					69	D5	00039	CALLS	#1, LIB\$SIGNAL		
					29	12	0003B	TSTL	CTLSGL_DCLPRSOWN	1886	
1C		00	6E		00	2C	0003D	BNEQ	3\$		
				E4	AD		00042	MOVCS	#0, (SP), #0, #28, REQ_DESC	1888	
				E4	AD	94	00044	CLRB	REQ_DESC	1889	
				04	AE	9F	00047	PUSHAB	REQ_FLAGS	1890	
				14	AE	9F	0004A	PUSHAB	RPW		
				E4	AD	9F	0004D	PUSHAB	REQ_DESC		
		00000000G	00		03	FB	00050	CALLS	#3, -SYSSCLI		
			58		50	D0	00057	MOVL	R0, STATUS		
			05		58	E8	0005A	BLBS	STATUS, 2\$		
					58	DD	0005D	PUSHL	STATUS	1891	
			6A		01	FB	0005F	CALLS	#1, LIB\$SIGNAL		
			69	14	AE	D0	00062	MOVL	RPW+4, CTLSGL_DCLPRSOWN	1892	
			50		69	D0	00066	MOVL	CTLSGL_DCLPRSOWN, WRK	1898	
			56	00000000G	00	D0	00069	MOVL	CTLSGL_CLINTOWN, R6	1904	
			57	008C	C6	9E	00070	MOVAB	140(R6), R7		
			67		J1	88	00075	BISB2	#1, (R7)		
			51	F9B6	C0	9E	00078	MOVAB	-1610(R0), R1	1909	
		08	AE		51	D0	0007D	MOVL	R1, TOKEN		
			0C		03	0C	00081	PROBER	#3, #12, @TOKEN	1910	
					45	13	00086	BEQL	9\$		
05		08	BE		1C	ED	00088	CMPZV	#28, #4, @TOKEN, #5	1911	
					20	1A	0008E	BGTRU	6\$		
07		08	BE		18	ED	00090	CMPZV	#24, #4, @TOKEN, #7	1912	
					18	1A	00096	BGTRU	6\$		
00		08	BE		18	ED	00098	CMPZV	#24, #4, @TOKEN, #0	1913	
					01	1A	0009E	BGTRU	4\$		
						04	000A0	RET			
			51	BA	A0	D1	000A1	CMPL	-70(WRK), R1	1914	
					01	1E	000A5	BGEQU	5\$		
						04	000A7	RET			
			51	B6	A0	9E	000A8	MOVAB	-74(R0), R1	1915	
			51	BA	A0	D1	000AC	CMPL	-70(WRK), R1		
					01	1B	000B0	BLEQU	7\$		
						04	000B2	RET			
			08	AE	C6	A0	D0	000B3	MOVL	-58(WRK), TOKEN	1921
					07	13	000B8	BEQL	8\$	1922	

	08	BE		0A		03	0C	000BA	PROBER	#3, #10, @TOKEN	1923	
						0C	13	000BF	BEQL	9\$		
			08	AE	CA	A0	D0	000C1	8\$:	MOVL	-54(WRK), TOKEN	1926
						08	13	000C6	BEQL	10\$	1927	
	08	BE		0A		03	0C	000C8	PROBER	#3, #10, @TOKEN	1928	
						01	12	000CD	9\$:	BNEQ	10\$	
							04	000CF	RET			
				67		01	8A	000D0	10\$:	BICB2	#1, (R7)	1934
					008D	C6	B4	000D3	CLRW	141(R6)	1941	
08		00		6E		00	2C	000D7	MOVCS	#0, (SP), #0, #8, 132(R6)	1942	
					0084	C6		000DC				
			7C	A6	04	AC	7D	000DF	MOVQ	GET_VM, 124(R6)	1947	
				67		02	8A	000E4	BICB2	#2, (R7)	1953	
1C		00		6E		00	2C	000E7	MOVCS	#0, (SP), #0, #28, 64(R6)	1954	
					40	A6		000EC				
1C		00		6E		00	2C	000EE	MOVCS	#0, (SP), #0, #28, 92(R6)	1955	
					5C	A6		000F3				
0040	8F	00		6E		00	2C	000F5	MOVCS	#0, (SP), #0, #64, (R6)	1956	
						66		000FC				
					008F	C6	94	000FD	CLRB	143(R6)	1957	
					78	A6	D4	00101	CLRL	120(R6)	1958	
					0C	AE	D4	00104	CLRL	INDEX	1971	
				52		56	D0	00107	MOVL	R6, PLM	1972	
					08	AE	9F	0010A	PUSHAB	TOKEN	1974	
					10	AE	9F	0010D	PUSHAB	INDEX		
				6B		02	FB	00110	CALLS	#2, GET_PARAM_TOKEN		
				58		50	D0	00113	MOVL	R0, STATUS		
				53	0C	AE	D0	00116	MOVL	INDEX, R3	1978	
				35		58	E9	0011A	11\$:	BLBC	STATUS, 14\$	1976
			01	A2		53	90	0011D	MOVB	R3, 1(PLM)	1978	
				62		53	90	00121	MOVB	R3, (PLM)	1979	
			03	A2		53	90	00124	MOVB	R3, 3(PLM)	1980	
					08	AE	9F	00128	12\$:	PUSHAB	TOKEN	1982
					10	AE	9F	0012B	PUSHAB	INDEX		
				6B		02	FB	0012E	CALLS	#2, GET_PARAM_TOKEN		
				58		50	D0	00131	MOVL	R0, STATUS		
				0D		58	E9	00134	BLBC	STATUS, 13\$		
			08	AE		0C	C3	00137	SUBL3	#12, TOKEN, R0	1984	
				04		00	ED	0013C	CMPZV	#0, #4, 3(R0), #1	1985	
						E4	12	00142	BNEQ	12\$		
				53	0C	AE	D0	00144	13\$:	MOVL	INDEX, R3	1989
				53		01	83	00148	SUBB3	#1, R3, 2(PLM)		
	01	03	50			04	C0	0014D	ADDL2	#4, PLM	1990	
			A0			C8	11	00150	BRB	11\$	1976	
							04	00152	14\$:	RET	1994	
		02	A2									

; Routine Size: 339 bytes, Routine Base: DCL\$ZCODE + 000D

```
1995 1 GLOBAL ROUTINE dcl$present (rqdesc, rqwork, rqbits) =
1996 1
1997 1 ---
1998 1
1999 1 Determine if an entity is present on the command line.
2000 1
2001 1 Inputs:
2002 1
2003 1 rqdesc = Address of request descriptor data structure
2004 1 rqword, rqbits = ignored
2005 1
2006 1 Outputs:
2007 1
2008 1 Routine value:
2009 1
2010 1 success = cli$_present
2011 1 cli$_locpres
2012 1 cli$_defaulted
2013 1
2014 1 failure = cli$_absent
2015 1 cli$_negated
2016 1 cli$_locneg
2017 1
2018 1 All errors are signalled.
2019 1 ---
2020 1
2021 2 BEGIN
2022 2
2023 2 MAP
2024 2 rqdesc : REF BBLOCK;
2025 2
2026 2 GLOBAL REGISTER
2027 2 block=9: REF BBLOCK,
2028 2 number=10,
2029 2 type=11;
2030 2
2031 2 LOCAL
2032 2 keyword_array : VECTOR [2*(dcl_c_context+1)+1];
2033 2
2034 2
2035 2 Initialize CLINT if necessary.
2036 2
2037 2 IF .ctl$gl_clintown EQL 0
2038 2 THEN initialize (.rqdesc [int_l_getvm],
2039 2 .rqdesc [int_l_freevm]);
2040 2
2041 2
2042 2 Verify that valid entities were specified.
2043 2
2044 2 P return_if_error (verify_entities (rqdesc [int_w_entlen],
2045 2 keyword_array));
2046 2
2047 2
2048 2 If the entity is reserved then it is always present. If it is
2049 2 a parameter or qualifier, then check it out.
2050 2
2051 2 CASE .type FROM min_entity TO max_entity
```

```
! Address of entity descriptor block
! Parameter/qualifier number
! Entity type
```

```
! Keyword array
```

```
! If not yet initialized,
! then initialize parsing
```

```
! Verify all specified entities
```

```
! Process each entity type differently
```



```

: 452      2052 2 OF SET
: 453      2053 2 [reserved_entity]: RETURN cli$_present;
: 454      2054 2 [param_entity]: RETURN parameter_present (keyword_array [2]);
: 455      2055 2 [qual_entity]: RETURN qualifier_present (keyword_array [2]);
: 456      2056 2 TES;
: 457      2057 2
: 458      2058 1 END;

```

			0E00 00000	.ENTRY	DCL\$PRESENT, Save R9,R10,R11	: 1995
	5E	BC	AE 9E 00002	MOVAB	-68(SP), SP	: 2037
		00000000G	00 D5 00006	TSTL	CTL\$GL_CLINTOWN	
			0D 12 0000C	BNEQ	1\$	
	50	04	AC D0 0000E	MOVL	RQDESC, R0	: 2039
	7E	10	A0 7D 00012	MOVQ	16(R0), -(SP)	: 2038
	FE92	CF	02 FB 00016	CALLS	#2, INITIALIZE	
			5E DD 0001B 1\$:	PUSHL	SP	: 2045
	7E	04	AC 08 C1 0001D	ADDL3	#8, RQDESC, -(SP)	
		00000000V	02 FB 00022	CALLS	#2, VERIFY_ENTITIES	
			50 E9 00029	BLBC	STATUS, 6\$	
	02	01	5B CF 0002C	CASEL	TYPE, #1, #2	: 2055
	0006	0019	000E 00030 2\$:	.WORD	4\$-2\$,-	
					5\$-2\$,-	
					3\$-2\$	
					#CLIS_PRESENT, R0	
			50 00000000G 8F D0 00036 3\$:	MOVL		
			04 0003D	RET		
			08 AE 9F 0003E 4\$:	PUSHAB	KEYWORD_ARRAY+8	: 2054
	00000000V	EF	01 FB 00041	CALLS	#1, PARAMETER_PRESENT	
			04 00048	RET		: 2055
			08 AE 9F 00049 5\$:	PUSHAB	KEYWORD_ARRAY+8	
	00000000V	EF	01 FB 0004C	CALLS	#1, QUALIFIER_PRESENT	
			04 00053 6\$:	RET		: 2058

; Routine Size: 84 bytes, Routine Base: DCL\$ZCODE + 0160

```

460 2059 1 ROUTINE parameter_present (keyword_list) : entity_linkage =
461 2060 1
462 2061 1 ---
463 2062 1
464 2063 1 Determine if a parameter value is present.
465 2064 1
466 2065 1 Inputs:
467 2066 1
468 2067 1 keyword_list = Address of list of keyword descriptors
469 2068 1
470 2069 1 block = Address of parameter entity descriptor block
471 2070 1 number = Parameter number
472 2071 1 type = Always paramter
473 2072 1
474 2073 1 Outputs:
475 2074 1
476 2075 1 routine value = status indicating presence
477 2076 1
478 2077 1 ---
479 2078 1
480 2079 2 BEGIN
481 2080 2
482 2081 2 MAP
483 2082 2 keyword_list : REF VECTOR;
484 2083 2
485 2084 2 EXTERNAL REGISTER
486 2085 2 block=9: REF BBLOCK, ! Address of descriptor block
487 2086 2 number=10, ! Parameter number
488 2087 2 type=11; ! Entity type (param_entity)
489 2088 2
490 2089 2 BIND
491 2090 2 wrk = ctl$gl_dclprson : REF BBLOCK, ! Address of command wrk area
492 2091 2 prmlim = ctl$gl_clintown [dcl_l_prm[im] : VECTOR; ! Parameter context array
493 2092 2
494 2093 2 LOCAL
495 2094 2 default, ! Default values flag
496 2095 2 plm : REF BBLOCK; ! Address of parameter limit
497 2096 2
498 2097 2
499 2098 2 Set parameter state variables
500 2099 2
501 2100 2 plm = prmlim [.number-1]; ! Find limits of this parameter
502 2101 2 ctl$gl_clintown [dcl_b_param] = .number; ! Save last parameter # requested
503 2102 2 ! (for local qualifier search)
504 2103 2
505 2104 2
506 2105 2 If the parameter is not explicitly present, then check to see if it
507 2106 2 has a default value or is present by default. If not, return CLIS_ABSENT.
508 2107 2
509 2108 2 IF .plm [plm_b_fstdesc] EQL 0 ! If parameter is missing
510 2109 2 THEN IF (.block [ent_w_defval] EQL 0) AND ! And has no default value
511 2110 2 NOT .block [ent_v_deftrue] ! And it is not present by default
512 2111 2 THEN RETURN clis_absent ! Then indicate not present
513 2112 2 ELSE default = true ! Else set default flag
514 2113 2 ELSE default = false; ! Else clear default flag
515 2114 2
516 2115 2 !

```



```

: 517 2116 2 ! The parameter is either present or defaulted. Now it's time to check
: 518 2117 2 ! for keywords. If a keyword list is specified, then call process_keyword_list
: 519 2118 2 ! to check for their presence.
: 520 2119 2
: 521 2120 2 IF .keyword_list [0] NEQ 0 ! If we have a keyword list
: 522 2121 3 THEN BEGIN
: 523 2122 3 LOCAL found, qual, token;
: 524 2123 3 qual = 0; ! Assume first token will be defaulted
: 525 2124 3 token = token_desc (.plm [plm_b_fstdesc]); ! Get first parameter token
: 526 2125 3 found = process_keyword_list (.block, keyword_list [0], ! Process the keyword list
: 527 2126 3 .token, .default, param_entity, 0, qual);
: 528 2127 3 IF .qual GTR 0 ! If value was not defaulted
: 529 2128 3 THEN plm [plm_b_quadesc] = table_index (.qual) + 1 ! Then update local qualifier context
: 530 2129 3 ELSE plm [plm_b_quadesc] = .plm [plm_b_lstdesc] + 1; ! Else allow no more local qualifiers
: 531 2130 3 RETURN .found;
: 532 2131 2 END;
: 533 2132 2
: 534 2133 2 ! If we've gotten this far, then no keywords are present and the specified
: 535 2134 2 ! parameter is either present or defaulted. Return the appropriate value.
: 536 2135 2
: 537 2136 2 IF .default ! If the parameter was defaulted
: 538 2137 2 THEN RETURN cli$_defaulted ! Then so indicate
: 539 2138 2 ELSE BEGIN
: 540 2139 3 plm [plm_b_quadesc] = .plm [plm_b_fstdesc] + 1; ! Update qualifier pointer
: 541 2140 3 RETURN cli$_present; ! Return present
: 542 2141 3 END;
: 543 2142 2
: 544 2143 2
: 545 2144 1 END;

```

001C 00000 PARAMETER PRESENT:

					WORD	Save R2,R3,R4	2059
		54	00000000G	00 9E 00002	MOVAB	WRK, R4	
		5E		04 C2 00009	SUBL2	#4, SP	
		50	00000000G	00 D0 0000C	MOVL	CTL\$GL CLINTOWN, R0	2091
		52	FC A04A	DE 00013	MOVAL	-4(R0)[NUMBER], PLM	2100
008F		C0		5A 90 00018	MOVB	NUMBER, 143(R0)	2101
			01	A2 95 0001D	TSTB	1(PLM)	2108
				17 12 00020	BNEQ	2\$	
			1C	A9 B5 00022	TSTW	28(BLOCK)	2109
				0D 12 00025	BNEQ	1\$	
08	04	A9		02 E0 00027	BBS	#2, 4(BLOCK), 1\$	2110
		50	00000000G	8F D0 0002C	MOVL	#CLI\$_ABSENT, R0	2111
				04 00033	RET		
		53		01 D0 00034	MOVL	#1, DEFAULT	2112
				02 11 00037	BRB	3\$	2109
				53 D4 00039	CLRL	DEFAULT	2113
			04	BC D5 0003B	TSTL	@KEYWORD_LIST	2120
				42 13 0003E	BEQL	5\$	
				6E D4 00040	CLRL	QUAL	2123
		50		64 D0 00042	MOVL	WRK, R0	2124
		51	01	A2 9A 00045	MOVZBL	1(PLM), R1	
		51		0C C4 00049	MULL2	#12, R1	

		50	F9AA	C140	9E	0004C	MOVAB	-1622(R1)[R0], TOKEN	:	
					5E	DD 00052	PUSHL	SP	:	2125
		7E			01	7D 00054	MOVQ	#1, -(SP)	:	
					09	BB 00057	PUSHR	#*M<R0,R3>	:	2126
			04	AC	DD 00059	PUSHL	KEYWORD_LIST	:	2125	
					59	DD 0005C	PUSHL	BLOCK	:	
		00000000V	EF		07	FB 0005E	CALLS	#7, PROCESS_KEYWORD_LIST	:	
					6E	D5 00065	TSTL	QUAL	:	2127
					12	15 00067	BLEQ	4\$	:	
	51		6E		64	C3 00069	SUBL3	WRK, QUAL, R1	:	2128
				064A	C1	9E 0006D	MOVAB	1610(R1), R1	:	
			51		0C	C6 00072	DIVL2	#12, R1	:	
03	A2		51		02	81 00075	ADDB3	#2, R1, 3(PLM)	:	
					04	0007A	RET		:	
03	A2	02	A2		01	81 0007B	ADDB3	#1, 2(PLM), 3(PLM)	:	2129
					04	00081	RET		:	2130
			08		53	E9 00082	BLBC	DEFAULT, 6\$	:	2137
			50	00000000G	8F	D0 00085	MOVL	#CLIS_DEFAULTED, R0	:	2139
					04	0008C	RET		:	
03	A2	01	A2		01	81 0008D	ADDB3	#1, 1(PLM), 3(PLM)	:	2140
			50	00000000G	8F	D0 00093	MOVL	#CLIS_PRESENT, R0	:	2141
					04	0009A	RET		:	2144

; Routine Size: 155 bytes, Routine Base: DCL\$ZCODE + 01B4

```

547 2145 1 ROUTINE qualifier_present (keyword_list) : entity_linkage =
548 2146 1
549 2147 1 ---
550 2148 1
551 2149 1 Determine if a qualfier value is present.
552 2150 1
553 2151 1 Inputs:
554 2152 1
555 2153 1 keyword_list = Address of list of keyword descriptors
556 2154 1
557 2155 1 block = Address of qualifier entity descriptor block
558 2156 1 number = Qualifier number
559 2157 1 type = Always 'qual_entity'
560 2158 1
561 2159 1 Outputs:
562 2160 1
563 2161 1 routine value = status indicating presence
564 2162 1
565 2163 1 ---
566 2164 1
567 2165 2 BEGIN
568 2166 2
569 2167 2 MAP
570 2168 2 keyword_list : REF VECTOR;
571 2169 2
572 2170 2 EXTERNAL REGISTER
573 2171 2 block=9: REF BBLOCK, ! Address of descriptor block
574 2172 2 number=10, ! Parameter number
575 2173 2 type=11; ! Entity type (param_entity)
576 2174 2
577 2175 2 LOCAL
578 2176 2 default, ! Default values flag
579 2177 2 token : REF BBLOCK, ! Address of next token descriptor
580 2178 2 status;
581 2179 2
582 2180 2
583 2181 2 Search for a local occurrence of the qualifier. If none found, and
584 2182 2 the qualifier can be positioned globally, then search for a global
585 2183 2 occurrence.
586 2184 2
587 2185 2 token = local_qualifier (.block, .number); ! Search for local qualifier
588 2186 2 IF (.token EQL 0) AND .block [ent_v verb] ! If none, but allowed globally,
589 2187 2 THEN token = global_qualifier (.block, .number); ! Then search for global qualifier
590 2188 2
591 2189 2
592 2190 2 If no occurrence was found, check to see if it is present by default.
593 2191 2 If not, then return CLIS_ABSENT.
594 2192 2
595 2193 2 IF .token EQL 0 ! If no occurrence found,
596 2194 2 THEN IF NOT .block [ent_v deftrue] ! and not defaulted present
597 2195 2 AND NOT (.block [ent_v batdef] AND batch_job()) !
598 2196 2 THEN RETURN cli$_absent ! Then return not present
599 2197 2 ELSE default = true ! Else set default flag
600 2198 2 ELSE default = false; ! Else clear default flag
601 2199 2
602 2200 2
603 2201 2 ! The qualifier is either explicitly, or defaulted, present. If any keywords

```

003C 00000 QUALIFIER PRESENT:

PC	Instruction	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
----	-------------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

		02	11	00054	BRB	5\$	:	2194
		50	D4	00056	CLRL	DEFAULT	:	2198
		04	BC	D5 00058	TSTL	@KEYWORD_LIST	:	2204
			1F	13 0005B	BEQL	6\$	:	
			7E	7C 0005D	CLRQ	-(SP)	:	2206
			02	DD 0005F	PUSHL	#2	:	
			50	DD 00061	PUSHL	DEFAULT	:	2207
			52	DD 00063	PUSHL	TOKEN	:	
		04	AC	DD 00065	PUSHL	KEYWORD_LIST	:	2206
			59	DD 00068	PUSHL	BLOCK	:	
	00000000V		07	FB 0006A	CALLS	#7, PROCESS_KEYWORD_LIST	:	
			50	D1 00071	CMPL	STATUS, R4	:	2208
			38	13 00074	BEQL	11\$	:	
			50	D1 00076	CMPL	STATUS, R3	:	2209
			18	12 00079	BNEQ	9\$	:	
				04 0007B	RET		:	2210
			50	E9 0007C	BLBC	DEFAULT, 7\$	:	2216
			54	D0 0007F	MOVL	R4, R0	:	2217
				04 00082	RET		:	
	09		62	E1 00083	BBC	#20, (TOKEN), 8\$	:	2218
			50	D0 00087	MOVL	#CLIS_NEGATED, STATUS	:	2219
			03	11 0008E	BRB	9\$	:	
			50	D0 00090	MOVL	R5, STATUS	:	2220
00			04	1C ED 00093	CMPZV	#28, #4, (TOKEN), #0	:	2226
			14	13 00098	BEQL	11\$	:	
			55	D1 0009A	CMPL	STATUS, R5	:	2228
			08	12 0009D	BNEQ	10\$	:	
			50	D0 0009F	MOVL	#CLIS_LOCPRES, R0	:	2229
				04 000A6	RET		:	
			50	D0 000A7	MOVL	#CLIS_LOCNEG, R0	:	2230
			04	000AE	RET		:	2232

; Routine Size: 175 bytes, Routine Base: DCL\$ZCODE + 024F

```

636 2233 1 GLOBAL ROUTINE dcl$getvalue (rdesc, rqwork, rqbits) =
637 2234 1
638 2235 1 ---
639 2236 1
640 2237 1 This routine is called to obtain the next value
641 2238 1 associated with a named entity on the command line.
642 2239 1
643 2240 1 Inputs:
644 2241 1
645 2242 1 rdesc = Address of request descriptor data structure
646 2243 1 rqword, rqbits = ignored
647 2244 1
648 2245 1 Outputs:
649 2246 1
650 2247 1 Routine value:
651 2248 1
652 2249 1 success = CLIS_CONCAT - a value terminated by a plus was returned
653 2250 1 CLIS_COMMA - a value terminated by a comma was returned
654 2251 1 SSS_NORMAL - a value was returned (there may be more)
655 2252 1
656 2253 1 failure = 0 - there are no more values associated with the entity
657 2254 1
658 2255 1 All errors are signaled.
659 2256 1 ---
660 2257 1
661 2258 2 BEGIN
662 2259 2
663 2260 2 MAP
664 2261 2 rdesc : REF BBLOCK;
665 2262 2
666 2263 2 GLOBAL REGISTER
667 2264 2 block=9: REF BBLOCK, ! Address of entity descriptor block
668 2265 2 number=10, ! Parameter/qualifier number
669 2266 2 type=11; ! Entity type
670 2267 2
671 2268 2 LOCAL
672 2269 2 keyword_array : VECTOR [2*(dcl_c_context+1)+1]; ! Keyword array
673 2270 2
674 2271 2
675 2272 2 Initialize CLINT if necessary.
676 2273 2
677 2274 2 IF .ctl$gl_clintown EQL 0 ! If not yet initialized,
678 2275 2 THEN initialize (.rdesc [int_l_getvm], ! then initialize parsing
679 2276 2 .rdesc [int_l_freevm]);
680 2277 2
681 2278 2
682 2279 2 Verify that valid entities were specified.
683 2280 2
684 2281 2 P return_if_error (verify_entities (rdesc [int_w_entlen], ! Verify all specified entities
685 2282 2 keyword_array));
686 2283 2
687 2284 2
688 2285 2 Process each primary entity type differently.
689 2286 2
690 2287 2 CASE .type FROM min_entity TO max_entity ! Process each entity type differently
691 2288 2 OF SET
692 2289 2 [param_entity]: RETURN parameter_value(.block, .number, ! Process parameter

```



```

: 693      2290 2      keyword_array [2], rqdesc [int_w_entlen]);
: 694      2291 2      [qual_entity]: RETURN qualifier_value(.block, .number,
: 695      2292 2      keyword_array [2], rqdesc [int_w_entlen]);
: 696      2293 2      [reserved_entity]: RETURN reserved_value(.number,
: 697      2294 2      rqdesc [int_w_entlen]);
: 698      2295 2      TES
: 699      2296 2
: 700      2297 1 END;

```

```

                                OE04 00000      .ENTRY DCL$GETVALUE, Save R2,R9,R10,R11      : 2233
                                5E      BC AE 9E 00002      MOVAB -68(SP), SP
                                00000000G 00 D5 00006      TSTL CTL$GL_CLINTOWN      : 2274
                                50      04 OD 12 0000C      BNEQ 1$
                                7E      10 AC D0 0000E      MOVL RQDESC, R0      : 2276
                                FCF4 CF 02 FB 00016      MOVQ 16(R0), -(SP)      : 2275
                                52      04 AC 0E DD 0001B 1$: PUSHL SP      : 2282
                                00000000V EF 02 FB 00024      ADDL3 #8, RQDESC, R2
                                02      01 50 E9 0002B      PUSHL R2
                                0026 0016 5B CF 0002E 2$: CALLS #2, VERIFY_ENTITIES
                                0006 00032      CASEL TYPE, #1, #2      : 2293
                                52 DD 00038 3$: PUSHL R2      : 2290
                                0C AE 9F 0003A      PUSHAB KEYWORD_ARRAY+8
                                7E      EF 59 7D 0003D      MOVQ BLOCK, =(SP)      : 2289
                                00000000V 04 FB 00040      CALLS #4, PARAMETER_VALUE
                                0C      52 DD 00048 4$: RET      : 2293
                                7E      EF 59 7D 0004D      PUSHL R2      : 2292
                                00000000V 04 FB 00050      PUSHAB KEYWORD_ARRAY+8
                                04 00057      MOVQ BLOCK, =(SP)      : 2291
                                52 DD 00058 5$: CALLS #4, QUALIFIER_VALUE
                                5A DD 0005A      RET      : 2293
                                00000000V EF 02 FB 0005C      PUSHL R2      : 2294
                                04 00063 6$: PUSHL NUMBER      : 2293
                                CALLS #2, RESERVED_VALUE
                                RET      : 2297

```

; Routine Size: 100 bytes, Routine Base: DCL\$ZCODE + 02FE

```

702 2298 1 ROUTINE parameter_value (entity, param_number, keyword_list, retdesc) =
703 2299 1
704 2300 1 ---
705 2301 1
706 2302 1 This routine returns the next value in the list for
707 2303 1 a given parameter.
708 2304 1
709 2305 1 Inputs:
710 2306 1
711 2307 1 entity = Address of parameter descriptor block
712 2308 1 param_number = Parameter number
713 2309 1 keyword_list = Address of list of keyword descriptors
714 2310 1 retdesc = Address of return descriptor to receive value
715 2311 1
716 2312 1 Outputs:
717 2313 1
718 2314 1 retdesc = Next value string in list
719 2315 1
720 2316 1 routine value = status indicating presence of value
721 2317 1 ---
722 2318 1
723 2319 2 BEGIN
724 2320 2
725 2321 2 MAP
726 2322 2 entity : REF BBLOCK,
727 2323 2 keyword_list : REF VECTOR,
728 2324 2 retdesc : REF BBLOCK;
729 2325 2
730 2326 2 BIND
731 2327 2 wrk = ctl$gl_dclprstown : REF BBLOCK,
732 2328 2 prmlim = ctl$gl_clintown [dcl_l_prm[im] : VECTOR,
733 2329 2 entity_context = ctl$gl_clintown [dcl_l_entity] : VECTOR,
734 2330 2 token_context = ctl$gl_clintown [dcl_l_token] : VECTOR;
735 2331 2
736 2332 2 LOCAL
737 2333 2 found,
738 2334 2 plm: REF BBLOCK;
739 2335 2
740 2336 2
741 2337 2 Set initial conditions.
742 2338 2
743 2339 2 found = true;
744 2340 2 retdesc [dsc$w_length] = 0;
745 2341 2 ctl$gl_clintown [dcl_b_param] = .param_number;
746 2342 2
747 2343 2 ctl$gl_clintown [dcl_l_qual] = 0;
748 2344 2 plm = prmlim [.param_number - 1];
749 2345 2 ctl$gl_clintown [dcl_b_param] = .param_number;
750 2346 2
751 2347 2
752 2348 2 Find our place in the parameter value list. If the parameter does not
753 2349 2 appear on the command line, see if it is present by default.
754 2350 2
755 2351 2 IF .plm [plm_b_fstdesc] EQL 0
756 2352 4 THEN IF (.entity [ent_w_defval] EQL 0)
757 2353 3 AND NOT .entity [ent_v_deftrue])
758 2354 3 OR (.plm [plm_b_nxtdesc] GTRU

```

```

! Address of command work area (WRK block)
! Parameter context array
! Entity context array
! Token context array
! Value found flag
! Address of current parameter context entry
! Assume value will be found
! and that the value will be null
! Save last parameter # requested
! (for local qualifier search)
! Clear qualifier context
! Find limits for the parameter
! Update the local qualifier context
! If param not on command line
! Then if param is not defaulted
! Or if all default values have

```


759	2355	3			
760	2356	3	THEN BEGIN	.plm [plm_b_lstdesc])	! already been returned
761	2357	3	zero_context_arrays (0);		! Then no value is present
762	2358	3	RETURN cli\$_absent;		! Clear the context arrays
763	2359	3	END;		! Return no value present
764	2360	2			
765	2361	2			
766	2362	2	Update the context if necessary.		
767	2363	2			
768	2364	2	IF .entity_context [0] NEQ .entity		! If there has been a change in context
769	2365	3	THEN BEGIN		! Then update the context
770	2366	3	entity_context [0] = .entity;		! Set the current entity context
771	2367	3	token_context [0] = 0;		! Set the current token context
772	2368	3	zero_context_arrays (1);		! Clear the rest of the context arrays
773	2369	2	END;		
774	2370	2			
775	2371	2			
776	2372	2	If no keyword list is specified, then if the parameter is present by default,		
777	2373	2	or has a default value, then return that default value. Otherwise, return		
778	2374	2	the next parameter value and update the context.		
779	2375	2			
780	2376	2	IF .keyword_list [0] EQL 0		! If no keyword list is specified
781	2377	2	THEN IF .plm [plm_b_fstdesc] EQL 0		! Then, if the param value is defaulted
782	2378	4	THEN (IF (found = get_default_value		! Then, return that default value
783	2379	3	(.entity, 0, .refdesc)) EQL true		! And, if this is the last default parameter
784	2380	3	THEN plm [plm_b_nxtdesc] =		! Then so indicate
785	2381	3	.plm [plm_b_lstdesc] + 1)		
786	2382	3	ELSE BEGIN		! Else, return the explicit value
787	2383	3	LOCAL token;		
788	2384	3			
789	2385	3	IF .plm [plm_b_nxtdesc] GTRU .plm [plm_b_lstdesc]		! If all values have been returned,
790	2386	4	THEN BEGIN		! Then no value is present
791	2387	4	plm [plm_b_quadesc] =		! Set no more local qualifiers
792	2388	4	.plm [plm_b_nxtdesc];		
793	2389	4	RETURN cli\$_absent;		! Return no value found
794	2390	3	END;		
795	2391	3			
796	2392	3	token = token_desc (.plm [plm_b_nxtdesc]);		! Get the first value token
797	2393	3	get_specified_value (.token, .refdesc);		! Return the value it marks
798	2394	3	plm [plm_b_quadesc] = .plm [plm_b_nxtdesc] + 1;		! Set local qualifier pointer
799	2395	3			
800	2396	4	IF (found = get_explicit_value (token, 0))		! Find the next parameter value
801	2397	4	AND (.found NEQ true)		! (but not the first in the next list)
802	2398	3	THEN plm [plm_b_nxtdesc] = table_index (.token)		! If present, then point to it
803	2399	4	ELSE BEGIN		! Else indicate no more values
804	2400	4	plm [plm_b_nxtdesc] =		! Force next CLISGET_VALUE to fail
805	2401	4	.plm [plm_b_lstdesc] + 1;		
806	2402	4	RETURN true;		! But return that this value was found
807	2403	3	END;		
808	2404	3			
809	2405	3	zero_context_arrays (1);		! Clear the rest of the context arrays
810	2406	3	END		
811	2407	3			
812	2408	3			
813	2409	3	If a keyword list is specified, then call process_keyword_list to check		
814	2410	3	the keywords.		
815	2411	3			

```

: 816      2412 3  ELSE BEGIN
: 817      2413 3  LOCAL default, qual, token;
: 818      2414 3
: 819      2415 3  IF .plm [plm_b_fstdesc] EQL 0
: 820      2416 3  THEN default = true
: 821      2417 4  ELSE BEGIN
: 822      2418 4  default = false;
: 823      2419 4  token = token_desc (.plm [plm_b_fstdesc]);
: 824      2420 3  END;
: 825      2421 3
: 826      2422 3  qual = 0;
: 827      2423 3  found = process_keyword_list (.entity, keyword_list [0],
: 828      2424 3  .token, .default, param_entity, .ret_desc, qual);
: 829      2425 3
: 830      2426 3  IF .qual GTR 0
: 831      2427 3  THEN plm [plm_b_quadesc] = table_index (.qual) + 1
: 832      2428 3  ELSE plm [plm_b_quadesc] = .plm [plm_b_lstdesc] + 1;
: 833      2429 3
: 834      2430 2  END;
: 835      2431 2
: 836      2432 2  RETURN .found;
: 837      2433 1  END;

```

! A keyword list has been specified

! If the param value is defaulted present
! Then set the default flag
! Else,
! Clear the default flag
! Find the first param token

! Assume first token will be defaulted
! Process the keywords

! If value was not defaulted
! Then update local qualifier context
! Else allow no more local qualifiers

! Return status

OFFC 00000 PARAMETER VALUE:

	5B	00000000G	00	9E	00002	WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	: 2298
	5E		08	C2	00009	MOVAB	WRK, R11	
	50	00000000G	00	D0	0000C	SUBL2	#8, SP	
	57	40	A0	9E	00013	MOVL	CTL\$GL_CLINTOWN, R0	: 2328
	58	5C	A0	9E	00017	MOVAB	64(R0), R7	: 2329
	5A		01	D0	0001B	MOVAB	92(R0), R8	: 2330
		10	BC	B4	0001E	MOVL	#1, FOUND	: 2339
008F	C0	08	AC	90	00021	CLRW	@RETDESC	: 2340
		78	A0	D4	00027	MOVAB	PARAM_NUMBER, 143(R0)	: 341
51		08	AC	D0	0002A	CLRL	120(R0)	: 343
56	FC	A041	DE	0002F	MOVL	PARAM_NUMBER, R1	: 2344	
008F	C0	08	AC	90	00035	MOVAL	-4(R0)[R1], PLM	
		59	D4	00039	MOVAB	PARAM_NUMBER, 143(R0)	: 2345	
		01	A6	95	0003B	CLRL	R9	: 2351
		24	12	0003E	TSTB	1(PLM)		
		59	D6	00040	BNEQ	3\$		
	50	04	AC	D0	00042	INCL	R9	
		1C	A0	B5	00046	MOVL	ENTITY, R0	: 2352
			05	12	00049	TSTW	28(R0)	
	06	04	A0	02	E1	BNEQ	1\$	
		02	A6	66	91	BBC	#2, 4(R0), 2\$	: 2353
				0E	1B	CMPB	(PLM), 2(PLM)	: 2355
1C	00	6E	00	2C	00056	BLEQU	3\$	
			67		0005B	MOVCS	#0, (SP), #0, #28, (R7)	: 2357
1C	00	6E	00	2C	0005C			
			68		00061	MOVCS	#0, (SP), #0, #28, (R8)	
			4D	11	00062	BRB	7\$	: 2358
	04	AC	67	D1	00064	CMPL	(R7), ENTITY	: 2364

		67	L4	14	13	00068	BEQL	4\$		
				AC	D0	0006A	MOVL	ENTITY, (R7)		2366
18	00	6E		68	D4	0006E	CLRL	(R8)		2367
				00	2C	00070	MOVCS	#0, (SP), #0, #24, 4(R7)		2368
18	00	6E	04	A7		00075				
				00	2C	00077	MOVCS	#0, (SP), #0, #24, 4(R8)		
			04	A8		0007C				
			0C	BC	D5	0007E	4\$: TSTL	@KEYWORD_LIST		2376
				03	13	00081	BEQL	5\$		
		1E		0096	31	00083	BRW	12\$		
				5C	E9	00086	5\$: BLBC	R9, 6\$		2377
			10	AC	DD	00089	PUSHL	RETDSC		2379
				7E	D4	0008C	CLRL	-(SP)		
			04	AC	DD	0008E	PUSHL	ENTITY		
	00000000V	EF		03	FB	00091	CALLS	#3, GET_DEFAULT_VALUE		
		5A		50	D0	00098	MOVL	R0, FOUND		
		01		5A	D1	0009B	CMPL	FOUND, #1		
				7A	12	0009E	BNEQ	11\$		
66	02	A6		01	81	000A0	ADDB3	#1, 2(PLM), (PLM)		2381
				73	11	000A5	BRB	11\$		2378
	02	A6		66	91	000A7	6\$: CMPB	(PLM), 2(PLM)		2385
				0C	1B	000AB	BLEQU	8\$		
	03	A6		66	90	000AD	MOVB	(PLM), 3(PLM)		2388
		50	00000000G	8F	D0	000B1	7\$: MOVL	#CLIS_ABSENT, R0		2389
					04	000B8	RET			
		50		6B	D0	000B9	8\$: MOVL	WRK, R0		2392
		51		66	9A	000BC	MOVZBL	(PLM), R1		
		51		0C	C4	000BF	MULL2	#12, R1		
		6E	F9AA	C140	9E	000C2	MOVAB	-1622(R1)[R0], TOKEN		
			10	AC	DD	000C8	PUSHL	RETDSC		2393
			04	AE	DD	000CB	PUSHL	TOKEN		
03	A6	00000000V		02	FB	000CE	CALLS	#2, GET_SPECIFIED_VALUE		
		66		01	81	000D5	ADDB3	#1, (PLM), 3(PLM)		2394
				7E	D4	000DA	CLRL	-(SP)		2396
			04	AE	9F	000DC	PUSHAB	TOKEN		
	00000000V	EF		02	FB	000DF	CALLS	#2, GET_EXPLICIT_VALUE		
		5A		50	D0	000E6	MOVL	R0, FOUND		
		17		5A	E9	000E9	BLBC	FOUND, 9\$		
		01		5A	D1	000EC	CMPL	FOUND, #1		2397
				12	13	000EF	BEQL	9\$		
	50	6E		6B	C3	000F1	SUBL3	WRK, TOKEN, R0		2398
		50	064A	C0	9E	000F5	MOVAB	1610(R0), R0		
		50		0C	C6	000FA	DIVL2	#12, R0		
66		50		01	81	000FD	ADDB3	#1, R0, (PLM)		
				09	11	00101	BRB	10\$		
66	02	A6		01	81	00103	9\$: ADDB3	#1, 2(PLM), (PLM)		2401
		50		01	D0	00108	MOVL	#1, R0		2402
				04	0010B		RET			
18	00	6E		00	2C	0010C	10\$: MOVCS	#0, (SP), #0, #24, 4(R7)		2405
			04	A7		00111				
18	00	6E		00	2C	00113	MOVCS	#0, (SP), #0, #24, 4(R8)		
			04	A8		00118				
				56	11	0011A	11\$: BRB	16\$		2377
		05		59	E9	0011C	12\$: BLBC	R9, 13\$		2415
		52		01	D0	0011F	MOVL	#1, DEFAULT		2416
				12	11	00122	BRB	14\$		
				52	D4	00124	13\$: CLRL	DEFAULT		2418

50		6B	D0	00126	MOVL	WRK, R0	:	2419
51	01	A6	9A	00129	MOVZBL	1(PLM), P1	:	
51		0C	C4	0012D	MULL2	#12, R1	:	
50	F9AA	C140	9E	00130	MOVAB	-1622(R1)[R0], TOKEN	:	
	04	AE	D4	00136	CLRL	QUAL	:	2422
	04	AE	9F	00139	PUSHAB	QUAL	:	2423
	10	AC	DD	0013C	PUSHL	RETDESC	:	2424
		01	DD	0013F	PUSHL	#1	:	2423
		05	BB	00141	PUSHR	#*M<R0,R2>	:	2424
	0C	AC	DD	00143	PUSHL	KEYWORD_LIST	:	2423
	04	AC	DD	00146	PUSHL	ENTITY	:	
		07	FB	00149	CALLS	#7, PROCESS_KEYWORD_LIST	:	
		50	D0	00150	MOVL	R0, FOUND	:	
	04	AE	D5	00153	TSTL	QUAL	:	2426
		14	15	00156	BLF	15\$	:	
50	04	AE	6B	C3	SUBC	WRK, QUAL, R0	:	2427
		50	C0	9E	MOVAB	1610(R0), R0	:	
		50	0C	C6	DIVL2	#12, R0	:	
03	A6	50	02	81	ADDB3	#2, R0, 3(PLM)	:	
			06	11	BRB	16\$	:	
03	A6	02	A6	01	ADDB3	#1, 2(PLM), 3(PLM)	:	2428
		50	5A	D0	MOVL	FOUND, R0	:	2432
			04	00175	RET		:	2433

; Routine Size: 374 bytes, Routine Base: DCL\$ZCODE + 0362

```

839 2434 1 ROUTINE qualifier_value (entity, qual_number, keyword_list, retdesc) =
840 2435 1
841 2436 1 ---
842 2437 1
843 2438 1 Return the next value in a qualifier value list.
844 2439 1
845 2440 1 Inputs:
846 2441 1
847 2442 1 entity = Address of qualifier descriptor block
848 2443 1 qual_number = Qualifier number
849 2444 1 keyword_list = Address of list of keyword descriptors
850 2445 1 retdesc = Address of return descriptor to receive value
851 2446 1
852 2447 1 Outputs:
853 2448 1
854 2449 1 retdesc = Next value in list
855 2450 1
856 2451 1 routine value = status indicating presence of value
857 2452 1 ---
858 2453 1
859 2454 2 BEGIN
860 2455 2
861 2456 2 MAP
862 2457 2 entity : REF BBLOCK,
863 2458 2 keyword_list : REF VECTOR,
864 2459 2 retdesc : REF BBLOCK;
865 2460 2
866 2461 2 BIND
867 2462 2 entity_context = ctl$gl_clintown [dcl_l_entity] : VECTOR, ! Entity context array
868 2463 2 token_context = ctl$gl_clintown [dcl_l_token] : VECTOR, ! Token context array
869 2464 2 last_qual = ctl$gl_clintown [dcl_l_qual]; ! Last qualifier token
870 2465 2
871 2466 2 GLOBAL REGISTER
872 2467 2 block=9: REF BBLOCK, ! Address of descriptor block
873 2468 2 number=10, ! Qualifier number
874 2469 2 type=11; ! Entity type
875 2470 2
876 2471 2 LOCAL
877 2472 2 found, ! Value found flag
878 2473 2 token; ! Address of token descriptor
879 2474 2
880 2475 2
881 2476 2 Set initial conditions.
882 2477 2
883 2478 2 found = true; ! Assume value will be found
884 2479 2 retdesc [dsc$w_length] = 0; ! Assume the value will not be found
885 2480 2
886 2481 2
887 2482 2 Find the last occurrence of the qualifier on the command line. If the
888 2483 2 qualifier does not appear on the command line, see if it is present by
889 2484 2 default. If it isn't, return CLIS_ABSENT.
890 2485 2
891 2486 2 token = local_qualifier (.entity, .qual_number); ! Look for local occurrences
892 2487 2 IF (.token EQ 0) AND (.entity [ent_v_verb]) ! If none, but global allowed,
893 2488 2 THEN token = global_qualifier (.entity, .qual_number); ! Then look for global occurrences
894 2489 2 IF .token EQL 0 ! If still none,
895 2490 2 THEN IF NOT .entity [ent_v_deftrue] ! Then check for default presence

```

```

896      2491      3      AND NOT (.entity [ent_v_batdef] AND batch_job())
897      2492      3      THEN BEGIN
898      2493      3          last_qual = 0;
899      2494      3          zero_context_arrays (0);
900      2495      3          RETURN cli$_absent;
901      2496      3      END;
902      2497      3
903      2498      3
904      2499      3      If we are not doing the same qualifier as last time, then reset our context.
905      2500      3      Otherwise, use the old context.
906      2501      3
907      2502      3      IF (.entity_context [0] NEQ .entity) OR (.last_qual NEQ .token)
908      2503      3      THEN BEGIN
909      2504      3          token_context [0] = 0;
910      2505      3          entity_context [0] = .entity;
911      2506      3          last_qual = .token;
912      2507      3          zero_context_arrays (1);
913      2508      3      END;
914      2509      3
915      2510      3
916      2511      3      The qualifier is explicitly, or defaulted, present. If no keywords
917      2512      3      are specified, then return the explicit or default qualifier value.
918      2513      3
919      2514      3      IF .keyword_list [0] EQL 0
920      2515      3      THEN IF .token LEQ 0
921      2516      3          THEN RETURN get_default_value(.entity, 0, .retdesc)
922      2517      3          ELSE RETURN get_next_value(.token, .entity, 0, .retdesc)
923      2518      3
924      2519      3
925      2520      3      If a keyword list is specified, then process that list.
926      2521      3
927      2522      3      ELSE BEGIN
928      2523      3          LOCAL default;
929      2524      3          IF .token EQL 0
930      2525      3              THEN default = true;
931      2526      3              ELSE default = false;
932      2527      3          RETURN process_keyword_list (.entity, keyword_list [0],
933      2528      3              .token, .default, qual_entity, .retdesc, 0);
934      2529      3      END;
935      2530      3
936      2531      3      END;

```

OFFC 00000 QUALIFIER VALUE:

50	00000000G	00	D0	00002	WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	: 2434
58	40	A0	9E	00009	MOVL	CTL\$GL_CLINTOWN, R0	: 2462
59	5C	A0	9E	0000D	MOVAB	64(R0), R8	: 2463
57	78	A0	9E	00011	MOVAB	92(R0), R9	: 2464
50		01	D0	00015	MOVL	#1, FOUND	: 2478
	10	BC	B4	00018	CLRW	@RETDESC	: 2479
	08	AC	DD	0001B	PUSHL	QUAL_NUMBER	: 2486
56	04	AC	D0	0001E	MOVL	ENTITY, R6	: 2486
		56	DD	00022	PUSHL	R6	: 2486

	00000000V	EF		02	FB	00024	CALLS	#2, LOCAL_QUALIFIER	
		5A		50	D0	0002B	MOVL	R0, TOKEN	
				13	12	0002E	BNEQ	1\$	2487
		0F	05	A6	E9	00030	BLBC	5(R6), 1\$	
			08	AC	DD	00034	PUSHL	QUAL_NUMBER	2488
				56	DD	00037	PUSHL	R6	
	00000000V	EF		02	FB	00039	CALLS	#2, GLOBAL_QUALIFIER	
		5A		50	D0	00040	MOVL	R0, TOKEN	
				5B	D4	00043	CLRL	R11	2489
				5A	D5	00045	TSTL	TOKEN	
				2C	12	00047	BNEQ	3\$	
				5B	D6	00049	INCL	R11	
25	04	A6		02	E0	0004B	BBS	#2, 4(R6), 3\$	2490
0A	04	A6		03	E1	00050	BBC	#3, 4(R6), 2\$	2491
	00000000V	EF		00	FB	00055	CALLS	#0, BATCH_JOB	
		16		50	E8	0005C	BLBS	R0, 3\$	
				67	D4	0005F	CLRL	(R7)	2493
1C	00	6E		00	2C	00061	MOVCS	#0, (SP), #0, #28, (R8)	2494
				68		00066			
1C	00	6E		00	2C	00067	MOVCS	#0, (SP), #0, #28, (R9)	
				69		0006C			
				50	00000000G	8F	D0	0006D	2495
						04		00074	
				56		68	D1	00075	3\$: 2502
						05	12	00078	
				5A		67	D1	0007A	
						16	13	0007D	
						69	D4	0007F	4\$: 2504
				68		56	D0	00081	2505
				67		5A	D0	00084	2506
18	00	6E		00	2C	00087	MOVCS	#0, (SP), #0, #24, 4(R8)	2507
				04	A8	0008C			
18	00	6E		00	2C	0008E	MOVCS	#0, (SP), #0, #24, 4(R9)	
				04	A9	00093			
				0C	BC	D5	00095	5\$: TSTL	@KEYWORD_LIST
					24	12	00098	BNEQ	7\$
					5A	D5	0009A	TSTL	TOKEN
					0F	14	0009C	BGTR	6\$
				10	AC	DD	0009E	PUSHL	RETDESC
					7E	D4	000A1	CLRL	-(SP)
					56	DD	000A3	PUSHL	R6
	00000000V	EF		03	FB	000A5	CALLS	#3, GET_DEFAULT_VALUE	
					04	000AC	RET		
				10	AC	DD	000AD	6\$: PUSHL	RETDESC
					7E	D4	000B0	CLRL	-(SP)
					56	DD	000B2	PUSHL	R6
					5A	DD	000B4	PUSHL	TOKEN
	00000000V	EF		04	FB	000B6	CALLS	#4, GET_NEXT_VALUE	
					04	000BD	RET		2522
		05		5B	E9	000BE	BLRC	R11, 8\$	2524
		50		01	D0	000C1	MOVL	#1, DEFAULT	2525
				02	11	000C4	BRB	9\$	
				50	D4	000C6	CLRL	DEFAULT	2526
				7E	D4	000C8	CLRL	-(SP)	2527
				10	AC	DD	000CA	PUSHL	RETDESC
					02	DD	000CD	PUSHL	#2
					50	DD	000CF	PUSHL	DEFAULT

RPCLINT
V04-000

I 14
16-Sep-1984 00:26:36 VAX-11 Bliss-32 V4.0-742 Page 30
14-Sep-1984 12:15:33 DISK\$VMSMASTER:[DCL.SRC]RPCLINT.B32;1 (10)

	OC	5A	DD	000D1	PUSHL	TOKEN	:
		AC	DD	000D3	PUSHL	KEYWORD_LIST	: 2527
00000000V	EF	56	DD	000D6	PUSHL	R6	:
		07	FB	000D8	CALLS	#7, PROCESS_KEYWORD_LIST	: 2531
		04	000DF	RET			:

; Routine Size: 224 bytes, Routine Base: DCL\$ZCODE + 0408


```

938 2532 1 ROUTINE reserved_value (number, retdesc) =
939 2533 1
940 2534 1 ---
941 2535 1
942 2536 1 Return the value associated with a reserved entity name.
943 2537 1
944 2538 1 Inputs:
945 2539 1
946 2540 1 number = Reserved word number
947 2541 1 retdesc = Address of return buffer descriptor
948 2542 1
949 2543 1 Outputs:
950 2544 1
951 2545 1 retdesc = Value string.
952 2546 1
953 2547 1 routine value = status indicating presence of value
954 2548 1 ---
955 2549 1
956 2550 2 BEGIN
957 2551 2
958 2552 2 MAP
959 2553 2 retdesc : REF BBLOCK;
960 2554 2
961 2555 2 BIND
962 2556 2 wrk = ctl$gl_dclprsn : REF BBLOCK;
963 2557 2
964 2558 2 LOCAL
965 2559 2 token : REF BBLOCK;
966 2560 2 string : VECTOR [2];
967 2561 2
968 2562 2 CASE .number FROM 1 TO 2
969 2563 2 OF SET
970 2564 3 [1]: BEGIN
971 2565 3 token = wrk [wrk_g_result];
972 2566 3
973 2567 4 WHILE (.token [ptr_v_type] NEQ ptr_k_endline)
974 2568 3 DO token = .token + ptr_c_length;
975 2569 3
976 2570 3 string [0] = .token [ptr_v_offset];
977 2571 3 string [1] = wrk [wrk_g_buffer];
978 2572 3
979 2573 3 IF CH$RCHAR (.string [1]) EQL %('$')
980 2574 4 THEN BEGIN
981 2575 4 string [0] = .string [0] - 1;
982 2576 4 string [1] = .string [1] + 1;
983 2577 3 END;
984 2578 2 END;
985 2579 2
986 2580 3 [2]: BEGIN
987 2581 3 LOCAL verb : REF VECTOR [,BYTE];
988 2582 3 verb = wrk [wrk_l_verb];
989 2583 3 string [0] = MINO (.verb [0], 4);
990 2584 3 string [1] = verb [1];
991 2585 2 END;
992 2586 2
993 2587 2 TES;
994 2588 2

```

```

! Address of command work area
! Address of curent token descriptor
! String descriptor
! Based on reserved word number
! $LINE reserved word
! Start at first token descriptor
! Until end of command line
! then skip to next one
! Line length is offset to eol
! and set address of input buffer
! If line is preceeded with '$'
! then strip it off
! $VERB reserved word
! Get address of ASCII verb string
! Get length of verb
! Get address of verb

```

```

: 995      2589 2 retdesc [dsc$w_length] = .string [0];
: 996      2590 2 retdesc [dsc$a_pointer] = .string [1];
: 997      2591 2
: 998      2592 2 RETURN true;
: 999      2593 1 END;

```

! Return value string

0000 00000 RESERVED VALUE:										
		5E		08	C2	00002		WORD	Save nothing	2532
		51	00000000G	00	D0	00005		SUBL2	#8, SP	
	01	01	04	AC	CF	0000C		MOVL	WRK, R1	2565
		002E		0004		00011	1\$:	CASEL	NUMBER, #1, #1	2562
								.WORD	2\$-1\$,-	
		50	F9B6	C1	9E	00015	2\$:	MOVAB	-1610(R1), TOKEN	2565
04	60	04		1C	ED	0001A	3\$:	CMPZV	#28, #4, (TOKEN), #4	2567
				05	13	0001F		BEQL	4\$	
		50		0C	C0	00021		ADDL2	#12, TOKEN	2568
				F4	11	00024		BRB	3\$	
6E	01	A0		0C	00	EF	4\$:	EXTZV	#0, #12, 1(TOKEN), STRING	2570
			04	AE	C1	9E	0002C	MOVAB	-2926(R1), STRING+4	2571
		24	F492	BE	91	00032		CMPB	@STRING+4, #36	2573
				1E	12	00036		BNEQ	7\$	
				6E	D7	00038		DECL	STRING	2575
			04	AE	D6	0003A		INCL	STRING+4	2576
				17	11	0003D		BRB	7\$	2562
		50	BE	A1	D0	0003F	5\$:	MOVL	-66(R1), VERB	2582
		51		60	9A	00043		MOVZBL	(VERB), R1	2583
		04		51	91	00046		CMPB	R1, #4	
				03	1B	00049		BLEQU	6\$	
		51		04	D0	0004B		MOVL	#4, R1	
		6E		51	D0	0004E	6\$:	MOVL	R1, STRING	
	04	AE	01	A0	9E	00051		MOVAB	1(R0), STRING+4	2584
		50	08	AC	D0	00056	7\$:	MOVL	RETDESC, R0	2589
		60		6E	B0	0005A		MOVW	STRING, (R0)	
	04	A0	04	AE	D0	0005D		MOVL	STRING+4, 4(R0)	2590
		50		01	D0	00062		MOVL	#1, R0	2592
				04	00065			RET		2593

; Routine Size: 102 bytes, Routine Base: DCL\$ZCODE + 05B8

```

: 1001 2594 1 ROUTINE verify_entities (entity_desc, array) : entity_linkage =
: 1002 2595 1
: 1003 2596 1 ---
: 1004 2597 1
: 1005 2598 1 Verify that all the specified entities really exist.
: 1006 2599 1 Fill in the keyword descriptor array. Return the address
: 1007 2600 1 of the descriptor corresponding to the first entity.
: 1008 2601 1
: 1009 2602 1 Inputs:
: 1010 2603 1
: 1011 2604 1 entity_desc = Address of entity name descriptor
: 1012 2605 1 array = Address of keyword array
: 1013 2606 1
: 1014 2607 1 Outputs:
: 1015 2608 1
: 1016 2609 1 block = Address of first entity descriptor
: 1017 2610 1 type = First entity type
: 1018 2611 1 number = Parameter or qualifier number
: 1019 2612 1
: 1020 2613 1 routine value = True if found, else error code
: 1021 2614 1
: 1022 2615 1 If entity not found, an error is signaled.
: 1023 2616 1 ---
: 1024 2617 1
: 1025 2618 2 BEGIN
: 1026 2619 2
: 1027 2620 2 MAP
: 1028 2621 2 array : REF VECTOR;
: 1029 2622 2
: 1030 2623 2 BIND
: 1031 2624 2 wrk = ctl$gl_dclprstown : REF BBLOCK; ! Address of command work area
: 1032 2625 2
: 1033 2626 2 EXTERNAL REGISTER
: 1034 2627 2 block=9: REF BBLOCK, ! Address of descriptor block
: 1035 2628 2 number=10, ! Parameter/qualifier number
: 1036 2629 2 type=11; ! Entity type
: 1037 2630 2
: 1038 2631 2
: 1039 2632 2 If we don't have a valid set of tables to search, exit with failure
: 1040 2633 2
: 1041 2634 2 IF .ctl$gl_clintown [dcl_v_nowrkarea] ! If invalid tables
: 1042 2635 2 THEN RETURN msg$_noentity; ! Then exit with failure
: 1043 2636 2
: 1044 2637 2
: 1045 2638 2 Convert the linear keyword list into a more usable keyword array.
: 1046 2639 2
: 1047 2640 2 return_if_error (convert_keyword_list (.entity_desc, .array)); ! Convert into a keyword array
: 1048 2641 2
: 1049 2642 2
: 1050 2643 2 Find the first entity.
: 1051 2644 2
: 1052 2645 2 IF NOT (find_main_entity (array [0])) ! If we can't find the first entity
: 1053 2646 2 THEN return_if_error (guess_entity (array [0])); ! Then look around for it
: 1054 2647 2
: 1055 2648 2
: 1056 2649 2 If entity was successfully found, then verify the keyword list.
: 1057 2650 2

```



```

: 1065 2657 1 ROUTINE find_main_entity (name): entity_linkage =
: 1066 2658 1
: 1067 2659 1 |---
: 1068 2660 1 |
: 1069 2661 1 |         Locate a parameter, qualifier, or reserved entity by entity name
: 1070 2662 1 |         string and return the address of the entity block corresponding to
: 1071 2663 1 |         that entity.
: 1072 2664 1 |
: 1073 2665 1 | Inputs:
: 1074 2666 1 |
: 1075 2667 1 |         name = Address of entity name descriptor
: 1076 2668 1 |
: 1077 2669 1 | Outputs:
: 1078 2670 1 |
: 1079 2671 1 |         block = Address of entity block
: 1080 2672 1 |         type = Entity type
: 1081 2673 1 |         number = Parameter or qualifier number
: 1082 2674 1 |
: 1083 2675 1 |         routine value = True if found, else false
: 1084 2676 1 |
: 1085 2677 1 |         If entity not found, an error is signaled.
: 1086 2678 1 |---
: 1087 2679 1
: 1088 2680 2 BEGIN
: 1089 2681 2
: 1090 2682 2 MAP
: 1091 2683 2     name:          REF BBLOCK;
: 1092 2684 2
: 1093 2685 2 BIND
: 1094 2686 2     wrk = ctl$gl_dclprstown : REF BBLOCK;
: 1095 2687 2
: 1096 2688 2 EXTERNAL REGISTER
: 1097 2689 2     block=9:      REF BBLOCK,
: 1098 2690 2     number=10,
: 1099 2691 2     type=11;
: 1100 2692 2
: 1101 2693 2 LOCAL
: 1102 2694 2     entity_name: VECTOR [2],
: 1103 2695 2     buffer: VECTOR [32,BYTE],
: 1104 2696 2     ptr:          REF BBLOCK;
: 1105 2697 2
: 1106 2698 2 |
: 1107 2699 2 |     Uppcase the entity name string
: 1108 2700 2 |
: 1109 2701 2 entity_name [1] = buffer;
: 1110 2702 2 upcase (.name, entity_name);
: 1111 2703 2
: 1112 2704 2 |
: 1113 2705 2 |     Search parameter and qualifier lists for entity.
: 1114 2706 2 |
: 1115 2707 2 block = .wrk [wrk_l_proptr];
: 1116 2708 2 type = param_entity;
: 1117 2709 2
: 1118 2710 3 WHILE (true)
: 1119 2711 3 DO BEGIN
: 1120 2712 4     IF (find_entity (entity_name))
: 1121 2713 3     THEN RETURN true;

```

! Address of command work area

! Address of descriptor block
! Parameter/qualifier number
! Entity type

! Descriptor for above buffer
! Buffer for upcased entity label
! Pointer to scan reserved word table

! Point at buffer
! Uppcase the string

! Get address of first block
! Assume parameter entity will be found

! Search parameter and qualifier lists

! Search entity list
! Return true if found

```

: 1122 2714 3 IF .type EQL qual_entity
: 1123 2715 3 THEN EXITLOOP;
: 1124 2716 3 block = .wrk [wrk_l_quablk];
: 1125 2717 3 type = qual_entity;
: 1126 2718 3 END;
: 1127 2719 3
: 1128 2720 3
: 1129 2721 3 Search the list of reserved entity names
: 1130 2722 3
: 1131 2723 3 number = 1;
: 1132 2724 3 ptr = reserved_words;
: 1133 2725 3 type = reserved_entity;
: 1134 2726 3
: 1135 2727 3 WHILE (CH$RCHAR(.ptr) NEQ 0)
: 1136 2728 3 DO BEGIN
: 1137 2729 3 IF CH$EQL(.entity_name [0], .entity_name [1],
: 1138 2730 3 CH$RCHAR(.ptr), .ptr+1, 0)
: 1139 2731 3 THEN RETURN true;
: 1140 2732 3 ptr = CH$RCHAR_A(ptr) + .ptr;
: 1141 2733 3 number = .number + 1;
: 1142 2734 3 END;
: 1143 2735 3
: 1144 2736 3 RETURN false;
: 1145 2737 3
: 1146 2738 1 END;

```

```

! If qualifiers already searched,
! then exit the loop
! Get address of first qualifier block
! Indicate qualifier entity

! Start at first reserved word
! Point to beginning of table
! Assume entity will be found

! Until end of table

! If this is the entity requested

! then return success
! Skip to next reserved word
! Increment reserved word number

! Return unsuccessful

```

```

                                003C 00000 FIND_MAIN ENTITY:
                                .WORD Save R2,R3,R4,R5
                                MOVAB WRK, R5
                                SUBL2 #40, SP
                                MOVAB BUFFER, ENTITY_NAME+4
                                PUSHAB ENTITY_NAME
                                PUSHAB NAME
                                CALLS #2, UPCASE
                                MOVL WRK, R0
                                MOVL -58(R0), BLOCK
                                MOVL #1, TYPE
                                PUSHAB ENTITY_NAME
                                CALLS #1, FIND_ENTITY
                                BLBS R0, 4$
                                CMPL TYPE, #2
                                BEQL 2$
                                MOVL WRK, R0
                                MOVL -54(R0), BLOCK
                                MOVL #2, TYPE
                                BRB 1$
                                MOVL #1, NUMBER
                                MOVAB RESERVED_WORDS, PTR
                                MOVL #3, TYPE
                                TSTB (PTR)
                                BEQL 6$
                                MOVZBL (PTR), R0
                                CMPC5 ENTITY_NAME, @ENTITY_NAME+4, #0, R0, 1(PTR)

```


RPCLINT
V04-000

C 15
16-Sep-1984 00:26:36
14-Sep-1984 12:15:33

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCL.SRC]RPCLINT.B32;1
Page 37
(13)

	01	A4	0005E					
		04	12 00060		BNEQ	5\$		:
50		01	D0 00062	4\$:	MOVL	#1, R0		: 2731
			04 00065		RET			:
50		84	9A 00066	5\$:	MOVZBL	(PTR)+, R0		: 2732
54		50	C0 00069		ADDL2	R0, PTR		:
		5A	D6 0006C		INCL	NUMBER		: 2733
		E0	11 0006E		BRB	3\$		: 2727
		50	D4 00070	6\$:	CLRL	R0		: 2736
			04 00072		RET			: 2738

; Routine Size: 115 bytes, Routine Base: DCL\$ZCODE + 0679

```

: 1148 2739 1 ROUTINE verify_keywords (input_block, input_type, keyword_list) =
: 1149 2740 1
: 1150 2741 1 ---
: 1151 2742 1
: 1152 2743 1     Verify a keyword path from the already found main entity.
: 1153 2744 1     (FIND_MAIN_ENTITY should always be called before this routine.)
: 1154 2745 1
: 1155 2746 1     Inputs:
: 1156 2747 1
: 1157 2748 1         input_block = Address of entity descriptor block
: 1158 2749 1         input_type = Type of entity
: 1159 2750 1         keyword_list = Address of keyword name descriptor list
: 1160 2751 1
: 1161 2752 1     Outputs:
: 1162 2753 1
: 1163 2754 1         routine value = True if found, else error code
: 1164 2755 1
: 1165 2756 1         If entity is not found, an error is signaled.
: 1166 2757 1 ---
: 1167 2758 1
: 1168 2759 2 BEGIN
: 1169 2760 2
: 1170 2761 2 MAP
: 1171 2762 2     keyword_list :      REF BBLOCK;
: 1172 2763 2
: 1173 2764 2 GLOBAL REGISTER
: 1174 2765 2     block=9:  REF BBLOCK,           ! Address of entity descriptor block
: 1175 2766 2     number=10,           ! Parameter/qualifier number
: 1176 2767 2     type=11;           ! Entity type
: 1177 2768 2
: 1178 2769 2 LOCAL
: 1179 2770 2     status;
: 1180 2771 2
: 1181 2772 2 !
: 1182 2773 2 !     Set up registers for find_keyword_entity().
: 1183 2774 2
: 1184 2775 2 block = .input_block;           ! Set block to entity descriptor
: 1185 2776 2 type = .input_type;           ! Set type to entity type
: 1186 2777 2 number = 0;           ! Clear number
: 1187 2778 2
: 1188 2779 2 !
: 1189 2780 2 !     Return error if entity is a reserved entity.
: 1190 2781 2
: 1191 2782 2 IF .type EQL reserved_entity           ! Is entity a reserved entity?
: 1192 2783 2 THEN status = msg$_noentity           ! Yes, then error
: 1193 2784 2
: 1194 2785 2 !
: 1195 2786 2 !     Search down the keyword path name.  Validate each keyword entity block
: 1196 2787 2 !     along the way.
: 1197 2788 2
: 1198 2789 3 ELSE DO BEGIN
: 1199 2790 4     IF NOT (status = find_keyword_entity(.keyword_list)) ! Validate the current keyword
: 1200 2791 3     THEN EXITLOOP;           ! Exit if invalid
: 1201 2792 3     keyword_list = .keyword_list + 8;           ! Get the next keyword
: 1202 2793 3     END
: 1203 2794 2     UNTIL (.keyword_list [dsc$_length] EQL 0);           ! Quit when no more keywords
: 1204 2795 2

```



```

: 1205      2796 2 |
: 1206      2797 2 | Signal any errors.
: 1207      2798 2 |
: 1208      2799 2 | IF NOT .status
: 1209      2800 2 |     THEN SIGNAL (msg$_noentity,1,.keyword_list,cli$_entnf);
: 1210      2801 2 |     RETURN .status;
: 1211      2802 2 |
: 1212      2803 1 | END;

```

! If some keyword was invalid
! Then signal the error
! Return the status

```

                                OEOC 00000 VERIFY_KEYWORDS:
                                .WORD Save R2,R3,R9,R10,R11
59      04 AC D0 00002      MOVL INPUT_BLOCK, BLOCK      : 2739
5B      08 AC D0 00006      MOVL INPUT_TYPE, TYPE      : 2775
                                5A D4 0000A      CLRL NUMBER      : 2776
03      5B D1 0000C      CMPL TYPE, #3      : 2777
                                09 12 0000F      BNEQ 1$      : 2782
53 000310FC      8F D0 00011      MOVL #200956, STATUS      : 2783
                                1F 11 00018      BRB 3$
52      0C AC D0 0001A 1$:      MOVL KEYWORD_LIST, R2      : 2790
                                52 DD 0001E 2$:      PUSHL R2
00000000V EF      01 FB 00020      CALLS #1, FIND_KEYWORD_ENTITY
53      50 D0 00027      MOVL R0, STATUS
                                53 E9 0002A      BLBC STATUS, 4$
                                0C AC      08 C0 0002D      ADDL2 #8, KEYWORD_LIST      : 2792
52      0C AC D0 00031      MOVL KEYWORD_LIST, R2      : 2794
                                62 B5 00035      TSTW (R2)
                                E5 12 00037      BNEQ 2$
18      53 E8 00039 3$:      BLBS STATUS, 5$      : 2799
                                8F DD 0003C 4$:      PUSHL #CLIS_ENTNF      : 2800
                                0C AC DD 00042      PUSHL KEYWORD_LIST
                                01 DD 00045      PUSHL #1
                                000310FC      8F DD 00047      PUSHL #200956
00000000G 00      04 FB 0004D      CALLS #4, LIB$SIGNAL
50      53 D0 00054 5$:      MOVL STATUS, R0      : 2801
                                04 00057      RET      : 2803

```

; Routine Size: 88 bytes, Routine Base: DCL\$ZCODE + 06EC

```

1214 2804 1 ROUTINE find_keyword_entity (keyword) : entity_linkage =
1215 2805 1
1216 2806 1 ---
1217 2807 1
1218 2808 1     Locate the keyword entity block specified by the next keyword
1219 2809 1     descriptor in the list.
1220 2810 1
1221 2811 1     Inputs:
1222 2812 1
1223 2813 1     keyword = Address of keyword name descriptor
1224 2814 1
1225 2815 1     Outputs:
1226 2816 1
1227 2817 1     block = Address of entity descriptor
1228 2818 1     type = Entity type
1229 2819 1     number = Parameter or qualifier number
1230 2820 1
1231 2821 1     routine value = True if found, else error code
1232 2822 1
1233 2823 1 ---
1234 2824 1
1235 2825 2 BEGIN
1236 2826 2
1237 2827 2 EXTERNAL REGISTER
1238 2828 2     block=9: REF BBLOCK,           ! Address of descriptor block
1239 2829 2     number=10,                   ! Keyword number
1240 2830 2     type=11;                   ! Entity type
1241 2831 2
1242 2832 2 BIND
1243 2833 2     wrk = ctl$gl_dclprstown : REF BBLOCK;           ! Address of command work area
1244 2834 2
1245 2835 2 LOCAL
1246 2836 2     buffer : BBLOCK [32],           ! Buffer for upcased keyword label
1247 2837 2     keyword_name : BBLOCK [dsc$c_s_bln];           ! Descriptor for keyword label
1248 2838 2
1249 2839 2
1250 2840 2     Get and upcase the keyword name string.
1251 2841 2
1252 2842 2     keyword_name [dsc$a_pointer] = buffer;           ! Point at buffer
1253 2843 2     upcase (.keyword, keyword_name);           ! Upcase the string
1254 2844 2
1255 2845 2
1256 2846 2     Get the first keyword entity block. If none, then exit with error.
1257 2847 2     If successful, then calculate the block address and search for the keyword.
1258 2848 2
1259 2849 2 IF .block [ent_l_user_type] EQL 0           ! Get the first keyword entity block
1260 2850 2     THEN RETURN msg$_noentity;           ! If none, then exit with an error
1261 2851 2     block = .block [ent_l_user_type] + .wrk [wrk_l_tab_vec];           ! Calculate its absolute address
1262 2852 2     block = .block [ent_l_next] + .wrk [wrk_l_tab_vec];           ! Skip list header
1263 2853 2     RETURN find_entity (.keyword_name);           ! Search for the keyword
1264 2854 1 END;

```

0000 00000 FIND_KEYWORD_ENTITY:

	04	5E		28	C2	00002	.WORD	Save nothing	:	2804
		AE	08	AE	9E	00005	SUBL2	#40, SP	:	
				5E	DD	0000A	MOVAB	BUFFER, KEYWORD_NAME+4	:	2842
			04	AC	DD	0000C	PUSHL	SP	:	2843
00000000V		EF		02	7B	0000F	PUSHL	KEYWORD	:	
			10	A9	D5	00016	CALLS	#2, UPCASE	:	
				08	12	00019	TSTL	16(BLOCK)	:	2849
		50	000310FC	8F	D0	0001B	BNEQ	1\$	:	
				04	0C	0022	MOVL	#200956, R0	:	2850
		50	00000000G	00	D0	00023	RET		:	
59	10	A9	DE	A0	C1	0002A	MOVL	WRK, R0	:	2851
59	08	A9	DE	A0	C1	00030	ADDL3	-34(R0), 16(BLOCK), BLOCK	:	
				5E	DD	00036	ADDL3	-34(R0), 8(BLOCK), BLOCK	:	2852
00000000V		EF		01	FB	00038	PUSHL	SP	:	2853
				04	00	003F	CALLS	#1, FIND_ENTITY	:	
							RET		:	2854

. Routine Size: 64 bytes, Routine Base: DCL\$ZCODE + 0744

```

: 1266 2855 1 ROUTINE find_entity (name) : entity_linkage =
: 1267 2856 1
: 1268 2857 1 ---
: 1269 2858 1
: 1270 2859 1      Locate an entity by entity name string and return the address
: 1271 2860 1      of the entity block corresponding to that entity.
: 1272 2861 1
: 1273 2862 1      Inputs:
: 1274 2863 1
: 1275 2864 1          name = Address of entity name descriptor
: 1276 2865 1
: 1277 2866 1      Outputs:
: 1278 2867 1
: 1279 2868 1          block = Address of entity block
: 1280 2869 1          type = Entity type
: 1281 2870 1          number = Parameter or qualifier number
: 1282 2871 1
: 1283 2872 1          routine value = True if found, else error code
: 1284 2873 1 ---
: 1285 2874 1
: 1286 2875 2 BEGIN
: 1287 2876 2
: 1288 2877 2 MAP
: 1289 2878 2     name : REF BBLOCK;
: 1290 2879 2
: 1291 2880 2 BIND
: 1292 2881 2     wrk = cti$gl_dclprstown : REF BBLOCK;           ! Address of command work area
: 1293 2882 2
: 1294 2883 2 EXTERNAL REGISTER
: 1295 2884 2     block=9: REF BBLOCK,           ! Address of descriptor block
: 1296 2885 2     number=10,           ! Parameter/qualifier number
: 1297 2886 2     type=11;           ! Entity type
: 1298 2887 2
: 1299 2888 2 LOCAL
: 1300 2889 2     label_string;
: 1301 2890 2
: 1302 2891 2     number = 1;           ! Start at entity 1
: 1303 2892 2
: 1304 2893 3 WHILE (.block NEQ 0)           ! Until end of entity list
: 1305 2894 2 DO
: 1306 2895 3     BEGIN
: 1307 2896 3
: 1308 2897 3     label_string = .block + .block [ent_w_label];       ! Get label address
: 1309 2898 3
: 1310 2899 3     IF (.name[dsc$w_length] EQL CH$RCHAR(.label_string)) AND ! Check length and 1st char..
: 1311 2900 4         (CH$RCHAR(.name[dsc$a_pointer]) EQL CH$RCHAR(.label_string+1))
: 1312 2901 3     THEN
: 1313 2902 4         BEGIN
: 1314 2903 4
: 1315 2904 4             IF CH$EQL(.name [dsc$w_length], .name [dsc$a_pointer], ! If this is the one we are looking for
: 1316 2905 4                 CH$RCHAR_A (.label_string), .label_string, 0)
: 1317 2906 4             THEN RETURN true;           ! then return success
: 1318 2907 3         END;
: 1319 2908 3         IF .block [ent_l_next] NEQ 0           ! If not end of list
: 1320 2909 3         THEN block = .block [ent_l_next] + .wrk [wrk_l_tab_vec] ! then skip to next block
: 1321 2910 3         ELSE RETURN msg$_noentity;           ! else terminate loop
: 1322 2911 3         number = .number + 1;           ! Increment entity number

```

```

: 1323      2912 2      END;
: 1324      2913 2
: 1325      2914 2 RETURN msg$_noentity;
: 1326      2915 1 END;

```

```

                                003C 00000 FIND_ENTITY:
                                .WORD
5A      01 D0 00002      Save R2,R3,R4,R5      : 2855
54      04 AC D0 00005      MOVL #1, NUMBER      : 2891
59      D5 00009 1$:      MOVL NAME, R4      : 2899
3C      13 0000B      TSTL BLOCK      : 2893
55      18 A9 3C 0000D      BEQL 3$      :
55      59 C0 00011      MOVZWL 24(BLOCK), LABEL_STRING      : 2897
50      65 9A 00014      ADDL2 BLOCK, LABEL_STRING      :
64      50 B1 00017      MOVZBL (LABEL_STRING), R0      : 2899
17      12 0001A      CMPW R0, (R4)      :
01 A5      04 B4 91 0001C      BNEQ 2$      :
10      12 00021      CMPB @4(R4), 1(LABEL_STRING)      : 2900
50      85 9A 00023      BNEQ 2$      :
50      64 2D 00026      MOVZBL (LABEL_STRING)+, R0      : 2905
74 B4      65      0002C      CMPC5 (R4), @4(R4), #0, R0, (LABEL_STRING)      : 2904
04      12 0002D      BNEQ 2$      :
50      01 D0 0002F      MOVL #1, R0      : 2906
04      04 00032      RET      :
08 A9      D5 00033 2$:      TSTL 8(BLOCK)      : 2908
11      13 00036      BEQL 3$      :
50 00000000G 00 D0 00038      MOVL WRK, R0      : 2909
59 08 A9      A0 C1 0003F      ADDL3 -34(R0), 8(BLOCK), BLOCK      :
SA      D6 00045      INCL NUMBER      : 2911
C0      11 00047      BRB 1$      : 2893
50 000310FC 8F D0 00049 3$:      MOVL #200956, R0      : 2914
04      04 00050      RET      : 2915

```

; Routine Size: 81 bytes, Routine Base: DCL\$ZCODE + 0784

```

1328 2916 1 ROUTINE guess_entity (array) : entity_linkage =
1329 2917 1
1330 2918 1 ---
1331 2919 1
1332 2920 1      Locate a given entity (or list of entities), that may be at any
1333 2921 1      level, by entity name string.  Fill in the array with the keyword
1334 2922 1      path that we found.  Search qualifiers first, then parameters.
1335 2923 1
1336 2924 1      Inputs:
1337 2925 1
1338 2926 1      array = Address of keyword array
1339 2927 1
1340 2928 1      Outputs:
1341 2929 1
1342 2930 1      block = Address of entity descriptor
1343 2931 1      type = Entity type
1344 2932 1      number = Parameter or qualifier number
1345 2933 1
1346 2934 1      routine value = True if found, else error code
1347 2935 1
1348 2936 1      If entity not found, an error is signaled.
1349 2937 1 ---
1350 2938 1
1351 2939 2 BEGIN
1352 2940 2
1353 2941 2 MAP
1354 2942 2     array : REF VECTOR;
1355 2943 2
1356 2944 2 BIND
1357 2945 2     wrk = ctl$gl_dclprstown : REF BBLOCK;           ! Address of command work area
1358 2946 2
1359 2947 2 EXTERNAL REGISTER
1360 2948 2     block=9: REF BBLOCK,           ! Address of descriptor block
1361 2949 2     number=10,                    ! Parameter/qualifier number
1362 2950 2     type=11;                     ! Entity type
1363 2951 2
1364 2952 2 LOCAL
1365 2953 2     found,
1366 2954 2     buffer : BBLOCK [32],           ! Buffer for upcased keyword label
1367 2955 2     keyword : BBLOCK [dsc$c_s_bln]; ! Descriptor for keyword label
1368 2956 2
1369 2957 2
1370 2958 2     Get and upcase the keyword name string.
1371 2959 2
1372 2960 2     keyword [dsc$a_pointer] = buffer;   ! Point at buffer
1373 2961 2     upcase (.array, keyword);           ! Upcase the string
1374 2962 2
1375 2963 2
1376 2964 2     Check the qualifier entity descriptors first.
1377 2965 2
1378 2966 2     block = .wrk [wrk_l_quablk];       ! Get first qualifier entity block
1379 2967 2     type = qual_entity;                ! Set qualifier type
1380 2968 2
1381 2969 2 WHILE (true)
1382 2970 2 DO BEGIN
1383 2971 2     number = 1;
1384 2972 2

```



```

1385 2973 4 WHILE (.block NEQ 0)           ! Check each entity in the list
1386 2974 4 DO BEGIN
1387 2975 4
1388 2976 5     IF (found = guess_keyword_entity (keyword, .block, 2,   ! Is keyword down this path?
1389 2977 5         array [0]))
1390 2978 4         THEN RETURN true;           ! Yes, then stop looking
1391 2979 4
1392 2980 4     IF .block [ent_l_next] NEQ 0
1393 2981 4         THEN block = .block [ent_l_next] + .wrk [wrk_l_tab_vec]; ! If more entity blocks
1394 2982 4         ELSE EXITLOOP;             ! Then calculate address of next
1395 2983 4                                     ! Else quit this loop
1396 2984 4     number = .number + 1;
1397 2985 4     END;
1398 2986 4
1399 2987 4
1400 2988 4     ! Check the parameter entity descriptors.
1401 2989 4
1402 2990 4     IF .type EQL param_entity
1403 2991 4         THEN EXITLOOP;
1404 2992 4     block = .wrk [wrk_l_proptr];
1405 2993 4     type = param_entity;
1406 2994 4     END;
1407 2995 4
1408 2996 4
1409 2997 4     ! If keyword was not found, then signal an error;
1410 2998 4
1411 2999 2 SIGNAL (msg$_noentity, 1, array [0], cli$_entnf);
1412 3000 2 RETURN msg$_noentity;
1413 3001 2
1414 3002 1 END;

```

```

0004 00000 GUESS_ENTITY:
      52 00000000G 00 9E 00002 .WORD Save R2
      5E 00000000 28 C2 00009 MOVAB WRK, R2
      04 AE 08 AE 9E 0000C SUBL2 #40, SP
      5E DD 00011 MOVAB BUFFER, KEYWORD+4
      04 AC DD 00013 PUSHL SP
      00000000V EF 02 FB 00016 PUSHL ARRAY
      50 62 D0 0001D CALLS #2, UPCASE
      59 CA A0 D0 00020 MOVL WRK, R0
      5B 02 D0 00024 MOVL -54(R0), BLOCK
      5A 01 D0 00027 MOVL #2, TYPE
      59 D5 0002A MOVL #1, NUMBER
      2A 13 CC02C TSTL BLOCK
      04 AC DD 0002E BEQL 4$
      02 DD 00031 PUSHL ARRAY
      59 DD 00033 PUSHL #2
      0C AE 9F 00035 PUSHL BLOCK
      00000000V EF 04 FB 00038 PUSHAB KEYWORD
      04 50 E9 0003F CALLS #4, GUESS_KEYWORD_ENTITY
      50 01 D0 00042 BLBC FOUND, 3$
      04 00045 MOVL #1, R0
      RET

```

RPCLINT
V04-000

L 15

16-Sep-1984 00:26:36

VAX-11 Bliss-32 V4.0-742

Page 46

14-Sep-1984 12:15:33

DISK\$VMSMASTER:[DCL.SRC]RPCLINT.B32;1 (17)

			08	A9	D5	00046	3\$:	TSTL	8(BLOCK)	:	2980
				0D	13	00049		BEQL	4\$	:	
		51		62	D0	0004B		MOVL	WRK, R1	:	2981
59	08	A9	DE	A1	C1	0004E		ADDL3	-34(R1), 8(BLOCK), BLOCK	:	
				5A	D6	00054		INCL	NUMBER	:	2984
				D2	11	00056		BRB	2\$	:	2973
		01		5B	D1	00058	4\$:	CMPL	TYPE, #1	:	2990
				0C	13	0005B		BEQL	5\$	:	
		51		62	D0	0005D		MOVL	WRK, R1	:	2992
		59	C6	A1	D0	00060		MOVL	-58(R1), BLOCK	:	
		5B		01	D0	00064		MOVL	#1, TYF	:	2993
				BE	11	00067		BRB	1\$	:	2969
			00000000G	8F	DD	00069	5\$:	PUSHL	#CLIS_ENTNF	:	2999
			04	AC	DD	0006F		PUSHL	ARRAY	:	
				01	DD	00072		PUSHL	#1	:	
			000310FC	8F	DD	00074		PUSHL	#200956	:	
00000000G		00		04	FB	0007A		CALLS	#4, LIB\$SIGNAL	:	
		50	000310FC	8F	D0	00081		MOVL	#200956, R0	:	3000
				04	00088			RET		:	3002

; Routine Size: 137 bytes, Routine Base: DCL\$ZCODE + 07D5


```

1416 3003 1 ROUTINE guess_keyword_entity (keyword, block, level, array) =
1417 3004 1
1418 3005 1 ---
1419 3006 1
1420 3007 1       Locate the keyword entity block specified by the next keyword
1421 3008 1       descriptor in the list.
1422 3009 1
1423 3010 1   Inputs:
1424 3011 1
1425 3012 1       name = Address of keyword name descriptor
1426 3013 1       block = Last entity block
1427 3014 1       level = Depth of search
1428 3015 1       array = Keyword descriptor array
1429 3016 1
1430 3017 1   Outputs:
1431 3018 1
1432 3019 1       array is initialized
1433 3020 1
1434 3021 1       routine value = True if found, else false
1435 3022 1
1436 3023 1       If entity is not found, an error is signaled.
1437 3024 1 ---
1438 3025 1
1439 3026 2 BEGIN
1440 3027 2
1441 3028 2 MAP
1442 3029 2     block : REF BBLOCK,
1443 3030 2     array : REF VECTOR,
1444 3031 2     keyword : REF BBLOCK;
1445 3032 2
1446 3033 2 BIND
1447 3034 2     wrk = ctl$gl_dclprstown : REF BBLOCK;           ! Address of command work area
1448 3035 2
1449 3036 2 LOCAL
1450 3037 2     keyword_label : REF VECTOR [,BYTE];           ! ASCII entity name from tables
1451 3038 2
1452 3039 2
1453 3040 2     Get the first keyword entity block.  If none, then exit with error.
1454 3041 2     If successful, then calculate the block address and continue.
1455 3042 2
1456 3043 3 IF (.block [ent_l_user_type] EQL 0)                  ! If no more keyword entity blocks
1457 3044 3     OR (.level EQL 2*(dcl_c_context + 1))           ! or too deeply nested
1458 3045 2     THEN RETURN false;                               ! Then exit with an error
1459 3046 2 block = .block [ent_l_user_type] + .wrk [wrk_l_tab_vec]; ! Calculate the address of the first block
1460 3047 2 block = .block [ent_l_next] + .wrk [wrk_l_tab_vec];   ! Skip list header
1461 3048 2
1462 3049 2
1463 3050 2     Starting with this block, search the keyword list for the specified keyword.
1464 3051 2
1465 3052 3 WHILE (.block NEQ 0)                                   ! Continue until an exitloop
1466 3053 3 DO BEGIN
1467 3054 3     keyword_label = .block + .block [ent_w_label];       ! Get keyword label
1468 3055 3
1469 3056 3
1470 3057 3     If we find the keyword here, then we are at the deepest point in
1471 3058 3     the search.  Shift the leftover keywords into the array at the
1472 3059 3     appropriate level.

```

```

: 1473 3060 3 !
: 1474 3061 3 IF CH$EQL (.keyword [dsc$w_length], .keyword [dsc$a_pointer], ! Is it the keyword we want
: 1475 3062 3 .keyword_label [0], keyword_label [1], 0)
: 1476 3063 4 THEN BEGIN ! Yes, then fill in the array
: 1477 3064 4 LOCAL t_level;
: 1478 3065 4 t_level = 1; ! Set level to start search at
: 1479 3066 4
: 1480 3067 6 WHILE ((.t_level LEQ dcl_c_context) ! Count number of levels of keywords
: 1481 3068 5 AND (.array [2 * .t_level] NEQ 0))
: 1482 3069 4 DO t_level = .t_level + 1;
: 1483 3070 4
: 1484 3071 4 IF (dcl_c_context - .level/2) LSSU .t_level ! If they won't fit in the leftover space
: 1485 3072 4 THEN RETURN false; ! Then exit with an error
: 1486 3073 4
: 1487 3074 4 CH$MOVE (4*2*.t_level, array [0], array [.level]); ! Shift the keywords over
: 1488 3075 4 RETURN true; ! Exit with success
: 1489 3076 3 END;
: 1490 3077 3
: 1491 3078 3 ! If the keyword is found further down the tree, then we are currently
: 1492 3079 3 at an intermediate level. Just insert the current keyword at the
: 1493 3080 3 appropriate level.
: 1494 3081 3
: 1495 3082 3 IF guess_keyword_entity (.keyword, .block, .level + 2, ! If the keyword is found deeper down
: 1496 3083 3 array [0])
: 1497 3084 4 THEN BEGIN
: 1498 3085 4 array [.level] = .keyword_label [0]; ! Then insert the current keyword
: 1499 3086 4 array [.level + 1] = keyword_label [1]; ! into the array
: 1500 3087 4 RETURN true; ! Exit with success
: 1501 3088 3 END;
: 1502 3089 3
: 1503 3090 3 !
: 1504 3091 3 ! If no match, but more blocks, then keep looking.
: 1505 3092 3
: 1506 3093 3 IF .block [ent_l_next] EQL 0 ! Are there more entity blocks?
: 1507 3094 3 THEN RETURN false; ! No, then error
: 1508 3095 3 block = .block [ent_l_next] + .wrk [wrk_l_tab_vec]; ! Yes, get next
: 1509 3096 2 END;
: 1510 3097 2
: 1511 3098 2 RETURN true; ! No-op
: 1512 3099 1 END;

```

OFFC 00000 GUESS_KEYWORD_ENTITY:

5B	00000000G	00	9E	00002	WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	: 3003
51	08	AC	D0	00009	MOVAB	WRK, R11	: 3043
	10	A1	D5	0000D	MOVL	BLOCK, R1	
		04	13	00010	TSTL	16(R1)	
10	0C	AC	D1	00012	BEQL	1\$	: 3044
		03	12	00016	CMPL	LEVEL, #16	
		00AC	31	00018	BNEQ	2\$	
		6B	D0	0001B	BRW	10\$	: 3046
08	AC	10	A1	DE	MOVL	WRK, R0	
		51	08	AC	ADDL3	-34(R0), 16(R1), BLOCK	: 3047
					MOVL	BLOCK, R1	

08	AC	08	A1	DE	A0	C1	00029	ADDL3	-34(R0), 8(R1), BLOCK	
			59	04	AC	D0	00030	MOVL	KEYWORD, R9	3061
			57	0C	AC	D0	00034	MOVL	LEVEL, R7	3082
			5A	02	A7	9E	00038	MOVAB	2(R7), R10	
			56	08	AC	D0	0003C	MOVL	BLOCK, R6	3052
					6D	13	00040	BEQL	7\$	
			58	18	A6	3C	00042	MOVZWL	24(R6), KEYWORD_LABEL	3054
			58		56	C0	00046	ADDL2	R6, KEYWORD_LABEL	
			50		68	9A	00049	MOVZBL	(KEYWORD_LABEL), R0	3062
50	00	04	B9	04	BC	2D	0004C	CMPC5	@KEYWORD, @4(R9), #0, R0, 1(KEYWORD_LABEL)	
				01	A8		00053			
					37	12	00055	BNEQ	6\$	
			50		01	D0	00057	MOVL	#1, T_LEVEL	3065
			07		50	D1	0005A	CMPL	T_LEVEL, #7	3067
					0E	14	0005D	BGTR	5\$	
	51		50		01	78	0005F	ASHL	#1, T_LEVEL, R1	3068
				10	BC41	D5	00063	TSTL	@ARRAY[R1]	
					04	13	00067	BEQL	5\$	
					50	D6	00069	INCL	T_LEVEL	3069
					ED	11	0006B	BRB	4\$	
			51	0C	AC	D0	0006D	MOVL	LEVEL, R1	3071
	52		51		02	C7	00071	DIVL3	#2, R1, R2	
			52		07	C2	00075	SUBL2	#7, R2	
			52		52	CE	00078	MNEGL	R2, R2	
			50		52	D1	0007B	CMPL	R2, T_LEVEL	
					47	1F	0007E	BLSSU	10\$	
			50		08	C4	00080	MULL2	#8, R0	3074
				10	BC41	DF	00083	PUSHAL	@ARRAY[R1]	
	9E	10	BC		50	28	00087	MOVC3	R0, @ARRAY, @ (SP)+	
					35	11	0008C	BRB	9\$	3075
				10	AC	D0	0008E	PUSHL	ARRAY	3083
				0440	8F	BB	00091	PUSHR	#*M<R6, R10>	
				04	AC	DD	00095	PUSHL	KEYWORD	
		FF63	CF		04	FB	00098	CALLS	#4, GUESS_KEYWORD_ENTITY	
			11		50	E9	0009D	BLBC	R0, 8\$	
		10	BC47		68	9A	000A0	MOVZBL	(KEYWORD_LABEL), @ARRAY[R7]	3085
			50	10	BC47	DE	000A5	MOVAL	@ARRAY[R7], R0	3086
		04	A0	01	A8	9E	000AA	MOVAB	1(KEYWORD_LABEL), 4(R0)	
					12	11	000AF	BRB	9\$	3087
				08	A6	D5	000B1	TSTL	8(R6)	3093
					11	13	000B4	BEQL	10\$	
			50		68	D0	000B6	MOVL	WRK, R0	3095
08	AC	08	A6	DE	A0	C1	000B9	ADDL3	-34(R0), 8(R6), BLOCK	
					FF79	31	000C0	BRW	3\$	3052
			50		01	D0	000C3	MOVL	#1, R0	3098
						04	000C6	RET		
					50	D4	000C7	CLRL	R0	3099
					04	000C9	RET			

; Routine Size: 202 bytes, Routine Base: DCL\$ZCODE + 085E

```

: 1514 3100 1
: 1515 3101 1 ROUTINE process_keyword_list (entity, keyword_list, token, default,
: 1516 3102 1 keyword_type, retdesc, qual) =
: 1517 3103 1 ---
: 1518 3104 1
: 1519 3105 1 Determine if the specified keywords are present.
: 1520 3106 1
: 1521 3107 1 Inputs:
: 1522 3108 1
: 1523 3109 1 entity = Address of primary entity block
: 1524 3110 1 keyword_list = Address of list of keyword descriptors
: 1525 3111 1 token = Address of first param value token or actual qualifier token
: 1526 3112 1 default = True if only checking default values (token is invalid)
: 1527 3113 1 keyword_type = Param_entity or qual_entity depending on type of keywords
: 1528 3114 1 retdesc = If specified, then we are doing a CLISGET_VALUE -
: 1529 3115 1 use and modify context and return the requested value
: 1530 3116 1 qual = Address of first possible address of qualifier token
: 1531 3117 1 (requested by parameter_present and parameter_value)
: 1532 3118 1
: 1533 3119 1 Outputs:
: 1534 3120 1
: 1535 3121 1 retdesc and qual are updated
: 1536 3122 1 routine value = status indicating presence
: 1537 3123 1
: 1538 3124 1 ---
: 1539 3125 1
: 1540 3126 2 BEGIN
: 1541 3127 2
: 1542 3128 2 MAP
: 1543 3129 2 entity : REF BBLOCK,
: 1544 3130 2 keyword_list : REF BBLOCK,
: 1545 3131 2 token : REF BBLOCK,
: 1546 3132 2 retdesc : REF BBLOCK;
: 1547 3133 2
: 1548 3134 2 BUILTIN
: 1549 3135 2 NULLPARAMETER;
: 1550 3136 2
: 1551 3137 2 GLOBAL REGISTER
: 1552 3138 2 block=9: REF BBLOCK,
: 1553 3139 2 number=10,
: 1554 3140 2 type=11;
: 1555 3141 2
: 1556 3142 2 BIND
: 1557 3143 2 entity_context = ctl$gl_clintown [dcl_l_entity] : VECTOR,
: 1558 3144 2 token_context = ctl$gl_clintown [dcl_l_token] : VECTOR,
: 1559 3145 2 last_qual = ctl$gl_clintown [dcl_l_qual];
: 1560 3146 2
: 1561 3147 2 LOCAL
: 1562 3148 2 continue,
: 1563 3149 2 ctx,
: 1564 3150 2 found,
: 1565 3151 2 get_value,
: 1566 3152 2 index,
: 1567 3153 2 negated,
: 1568 3154 2 plm : REF BBLOCK,
: 1569 3155 2 temp_token;
: 1570 3156 2

```

```

: Address of descriptor block
: Parameter number
: Entity type

: Entity context array
: Token context array
: Last qualifier token

: Local loop flag
: Index into context arrays
: Value found flag
: CLISGET_VALUE in progress flag
: Result parse descriptor number
: Explicit keyword negated flag
: Address of parameter limit
: Temporary token storage

```

```

1571 3157 2  |
1572 3158 2  | Set initial state variables.
1573 3159 2  |
1574 3160 2  | get_value = NOT NULLPARAMETER (6);
1575 3161 2  | ctx = 1;
1576 3162 2  | block = .entity;
1577 3163 2  | found = true;
1578 3164 2  | negated = false;
1579 3165 2  |
1580 3166 2  |
1581 3167 2  | Search for each keyword in the keyword path name.
1582 3168 2  |
1583 3169 4  | WHILE ((.keyword_list [dsc$w_length] NEQ 0)
1584 3170 3  |         AND .found)
1585 3171 3  | DO BEGIN
1586 3172 3  |     find_keyword_entity (.keyword_list);
1587 3173 3  |     keyword_list = .keyword_list + 8;
1588 3174 3  |
1589 3175 3  |
1590 3176 3  |     | If we are doing a CLISGET_VALUE, then we must concern ourselves with the old
1591 3177 3  |     | context arrays. If we are using a previous context, then update the token
1592 3178 3  |     | pointer, otherwise reset the context.
1593 3179 3  |
1594 3180 3  |     IF .get_value
1595 3181 3  |         THEN IF .block NEQ .entity_context [.ctx]
1596 3182 4  |             THEN zero_context_arrays (.ctx)
1597 3183 4  |             ELSE BEGIN
1598 3184 4  |
1599 3185 6  |                 IF ((.keyword_type EQL qual_entity)
1600 3186 5  |                     AND (.ctx EQL 1))
1601 3187 4  |                     THEN token = .last_qual
1602 3188 4  |                     ELSE token = .token_context [.ctx-1];
1603 3189 4  |
1604 3190 4  |                 IF .token LEQ 0
1605 3191 4  |                     THEN default = true
1606 3192 4  |                     ELSE negated = .token [ptr_v_negate];
1607 3193 4  |
1608 3194 3  |             END;
1609 3195 3  |
1610 3196 3  |
1611 3197 3  |     | If we have not yet encountered a defaulted keyword, then try to find
1612 3198 3  |     | the specified keyword either in the context or in the command string.
1613 3199 3  |
1614 3200 3  |     IF NOT .default
1615 3201 4  |         THEN BEGIN
1616 3202 4  |
1617 3203 4  |             LOCAL explicit;
1618 3204 4  |             explicit = true;
1619 3205 4  |
1620 3206 4  |             IF .get_value AND
1621 3207 5  |                 (.block EQL .entity_context [.ctx])
1622 3208 5  |             THEN BEGIN
1623 3209 5  |                 token = .token_context [.ctx-1];
1624 3210 5  |                 IF .token EQL 0
1625 3211 5  |                     THEN explicit = found = false;
1626 3212 4  |             END;
1627 3213 4  |

```

! True if CLISGET_VALUE, false if CLISPRESEN
! Start with the second array elements
! Start with primary entity block
! Assume the keyword/value will be found
! Assume no keyword will be negated

! While we have more keywords
! and are still successful

! Get the keyword entity block
! Point to the next keyword descriptor

! If we are doing a CLISGET_VALUE
! And if we have no valid previous context
! Then erase the old context
! Else try to use it

! If at /QUAL= level
! Then use the last qualifier context
! Else use the last token context

! If defaulted last time
! Then set default value flag
! Else conditionally set negated flag

! If no keyword was defaulted yet

! Assume some explicit value (not necessarily
! a match) is present

! If valid context exists

! Then use it

! Was it defaulted?
! Yes, then say so

1628	3214	6	IF ((.keyword_type EQL qual_entity)	! If qualifier value
1629	3215	5	OR (.ctx NEQ 1))	! or not level 1 parameter value,
1630	3216	5	AND NOT (.get_value AND	! Then if not using an old context
1631	3217	5	(.entity_context [.ctx] NEQ 0))	
1632	3218	4	THEN explicit = found =	! Then get the first value in the list
1633	3219	4	get_explicit_value (token, 1);	
1634	3220	4		
1635	3221	4	temp_token = 0;	! Init temporary token
1636	3222	4	continue = false;	! Init local flag
1637	3223	4		
1638	3224	5	WHILE (.found)	! Get last occurrence of keyword
1639	3225	5	DO BEGIN	
1640	3226	5		
1641	3227	7	WHILE (.continue OR (.found AND	! Check all keywords in the value list
1642	3228	6	(.token [ptr_b_number] NEQ .number)))	
1643	3229	6	DO BEGIN	
1644	3230	6	continue = false;	! for the one we want
1645	3231	6	found = get_explicit_value (token, 0);	
1646	3232	5	END;	
1647	3233	5		
1648	3234	5	IF .found	! If no keyword was found
1649	3235	6	THEN BEGIN	! Save found token
1650	3236	6	temp_token = .token;	
1651	3237	6	continue = true;	
1652	3238	6	END	
1653	3239	5	ELSE IF .temp_token NEQ 0	! Return found token
1654	3240	6	THEN BEGIN	
1655	3241	6	token = .temp_token;	
1656	3242	6	found = true;	
1657	3243	6	EXITLOOP;	
1658	3244	5	END;	
1659	3245	4	END;	
1660	3246	4		
1661	3247	4	IF .found	! If an explicit match was found
1662	3248	4	THEN negated = .token [ptr_v_negate]	! Then conditionally set negated flag
1663	3249	4	ELSE IF NOT .explicit	! If no match was found and no values were p
1664	3250	4	THEN default = true;	! Then plan to look for a default value
1665	3251	4		
1666	3252	3	END;	
1667	3253	3		
1668	3254	3		
1669	3255	3	! If some keyword was defaulted, or no explicit value was found, then	
1670	3256	3	check to see if this keyword was defaulted.	
1671	3257	3		
1672	3258	3	IF .default	! If no explicit value is present
1673	3259	3	THEN IF NOT .block [ent_v_deftrue]	! Then if the keyword is not defaulted prese
1674	3260	3	THEN found = false	! Then mark it not found
1675	3261	3	ELSE found = true;	! Else mark it found
1676	3262	3		
1677	3263	3		
1678	3264	3	! If we are doing a CLISGET_VALUE, and if we found the keyword, then update	
1679	3265	3	the context arrays.	
1680	3266	3		
1681	3267	4	IF (.get_value AND .found)	! If context should be updated
1682	3268	4	THEN BEGIN	! Then do so
1683	3269	4	IF NOT .default	! If keyword was explicitly found
1684	3270	4	THEN token_context [.ctx-1] = .token	! Then use that found token


```

: 1685      3271  4      ELSE IF .entity_context [.ctx] EQL 0      ! Else if no old context
: 1686      3272  4      THEN token_context [.ctx-1] = 0      ! Then mark keyword 'new default'
: 1687      3273  4      ELSE IF token_context [.ctx-1] LEQ 0      ! Else if already defaulted
: 1688      3274  4      THEN token_context [.ctx-1] = -1      ! Then mark 'default value already returned'
: 1689      3275  4      ELSE token_context [.ctx-1] = 0;      ! Else mark 'new default'
: 1690      3276  4      entity_context [.ctx] = .block;      ! Update the entity context
: 1691      3277  3      END;
: 1692      3278  3
: 1693      3279  3      !
: 1694      3280  3      ! Save the first possible address of a quaifier for the parameter_present and
: 1695      3281  3      ! parameter_value routines.
: 1696      3282  3
: 1697      3283  3      IF NOT NULLPARAMETER (7) AND (.ctx EQL 1) AND .found AND NOT .default
: 1698      3284  3      THEN .qual = .token;
: 1699      3285  3
: 1700      3286  3      ctx = .ctx + 1;      ! Increment the context level
: 1701      3287  2      END;
: 1702      3288  2
: 1703      3289  2      !
: 1704      3290  2      ! If no value was found, then return that status now.
: 1705      3291  2
: 1706      3292  2      IF NOT .found      ! If the keyword was not found
: 1707      3293  2      THEN RETURN cli$_absent;      ! Then say so
: 1708      3294  2
: 1709      3295  2      !
: 1710      3296  2      ! If we are doing a PRESENT, then return the correct found status now.
: 1711      3297  2
: 1712      3298  2      IF NULLPARAMETER (6)      ! If doing a PRESENT
: 1713      3299  2      THEN IF .default      ! If value was defaulted
: 1714      3300  2      THEN RETURN cli$_defaulted      ! Then say so
: 1715      3301  2      ELSE IF .token [ptr_v negate]      ! Else if it was negated
: 1716      3302  2      THEN RETURN cli$_negated      ! Then so indicate
: 1717      3303  2      ELSE RETURN cli$_present;      ! Else say it was present
: 1718      3304  2
: 1719      3305  2      !
: 1720      3306  2      ! If we are doing GET_VALUE then return the appropriate value and status.
: 1721      3307  2
: 1722      3308  2      IF .default      ! If the keyword was defaulted
: 1723      3309  2      THEN IF .negated      ! and if a keyword was negated
: 1724      3310  2      THEN RETURN cli$_absent      ! Then return absent
: 1725      3311  2      ELSE RETURN get_default_value(.block, .ctx-1, .retdesc);      ! Else return the default value
: 1726      3312  2      ELSE RETURN get_next_value(.token, .block, .ctx-1, .retdesc);      ! Else return the explicit value
: 1727      3313  2
: 1728      3314  1      END;

```

```

                                OFFC 00000 PROCESS_KEYWORD_LIST:
                                .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11      ; 3101
                                SUBL2 #32, SP      ;
                                MOVL CTL$GL_CLINTOWN, R0      ; 3143
                                MOVAB 64(R0), 8(SP)
                                MOVAB 62(R0), R8
                                MOVAB 120(R0), 4(SP)
                                CMPB (AP), #6      ; 3144
                                ; 3145
                                ; 3160

```

				57	05	1E	0001D	BGEQU	1\$		
					01	D0	0001F	MOVL	#1, R7		
					09	11	00022	BRB	2\$		
					57	D4	00024	CLRL	R7		
					18	AC	D5	00026	TSTL	24(AP)	
					02	12	00029	BNEQ	2\$		
					57	D6	0002B	INCL	R7		
				57	57	D2	0002D	MCOML	R7, GET_VALUE		
				56	01	D0	00030	MOVL	#1, CTX		3161
				59	04	AC	D0	00033	MOVL	ENTITY, BLOCK	3162
				AE	01	D0	00037	MOVL	#1, FOUND		3163
					18	AE	D4	0003B	CLRL	NEGATED	3164
					08	BC	B5	0003E	TSTW	@KEYWORD_LIST	3169
					03	03	12	00041	BNEQ	4\$	
					0183	31	00043	BRW	32\$		
				03	10	AE	E8	00046	BLBS	FOUND, 5\$	3170
					0183	31	0004A	BRW	37\$		
					08	AC	DD	0004D	PUSHL	KEYWORD_LIST	3172
					01	FB	00050	CALLS	#1, FIND_KEYWORD_ENTITY		
					08	C0	00055	ADDL2	#8, KEYWORD_LIST		3173
					57	E9	00059	BLBC	GET_VALUE, TOS		3180
					59	D1	0005C	CMPL	BLOCK, @8(SP)[CTX]		3181
					26	13	00061	BEQL	6\$		
					F9	A6	9E	00063	MOVAB	-7(R6), R0	3182
					04	C4	00067	MULL2	#4, R0		
					50	CE	0006A	MNEGL	R0, 12(SP)		
					08	BE46	DE	0006E	MOVAL	@8(SP)[CTX], (SP)	
					00	2C	00073	MOVC5	#0, (SP), #0, 12(SP), @0(SP)		
					00	BE	00079				
					6846	DE	0007B	MOVAL	(R8)[CTX], (SP)		
					00	2C	0007F	MOVC5	#0, (SP), #0, 12(SP), @0(SP)		
					00	BE	00085				
					27	11	00087	BRB	10\$		3181
					02	14	AC	D1	00089	6\$:	3185
						0C	12	0008D	BNEQ	7\$	
					01	56	D1	0008F	CMPL	CTX, #1	3186
						07	12	00092	BNEQ	7\$	
					0C	AC	04	BE	D0	00094	3187
						06	11	00099	BRB	8\$	
					0C	AC	FC	A846	D0	0009B	7\$:
						06	14	000A1	8\$:		3188
						01	D0	000A3	BGTR	9\$	3190
						07	11	000A7	MOVL	#1, DEFAULT	3191
						14	EF	000A9	BRB	10\$	
						AC	E9	000B0	EXTZV	#20, #1, @TOKEN, NEGATED	192
						00AA	31	000B4	BLBC	DEFAULT, 11\$	200
						01	D0	000B7	BRW	23\$	
						57	E9	000BA	MOVL	#1, EXPLICIT	3204
						59	D1	000BD	BLBC	GET_VALUE, 12\$	3206
						0D	12	000C2	CMPL	BLOCK, @8(SP)[CTX]	3207
						05	12	000C4	BNEQ	12\$	
						AE	D4	000CC	MOVL	-4(R8)[CTX], TOKEN	3209
						52	D4	000CF	BNEQ	12\$	3210
						AC	D1	000D1	CLRL	FOUND	3211
						05	13	000D5	CLRL	EXPLICIT	
						56	D1	000D7	CMPL	KEYWORD_TYPE, #2	3214
									BEQL	13\$	
									CMPL	CTX, #1	3215

			06	1D 13 000D1	BEQL 15\$		
				57 E9 000DC 13\$:	BLBC GET VALUE, 14\$		3216
			08 BE46	D5 000DF	TSTL @8(SP)[CTX]		3217
				14 12 000E3	BNEQ 15\$		
				01 DD 000E5 14\$:	PUSHL #1		3219
			0C AC	9F 000E7	PUSHAB TOKEN		
		00000000V	EF	02 FB 000EA	CALLS #2, GET EXPLICIT_VALUE		
		10	AE	50 D0 000F1	MOVL R0, FOUND		
			52	10 AE D0 000F5	MOVL FOUND, EXPLICIT		3220
				14 AE D4 000F9 15\$:	CLRL TEMP_TOKEN		3221
				1C AE D4 000FC	CLRL CONTINUE		3222
			53	10 AE E9 000FF 16\$:	BLBC FOUND, 21\$		3224
			10	1C AE E8 00103 17\$:	BLBS CONTINUE, 18\$		3227
			30	10 AE E9 00107	BLBC FOUND, 20\$		
			50	0C AC D0 0010B	MOVL TOKEN, R0		3228
5A	05	A0	08	00 ED 0010F	CMPZV #0, #8, 5(R0), NUMBER		
				15 13 00115	BEQL 19\$		
				1C AE D4 00117 18\$:	CLRL CONTINUE		3230
				7E D4 0011A	CLRL -(SP)		3231
			0C AC	9F 0011C	PUSHAB TOKEN		
		00000000V	EF	02 FB 0011F	CALLS #2, GET EXPLICIT_VALUE		
		10	AE	50 D0 00126	MOVL R0, FOUND		
			OB	D7 11 0012A	BRB 17\$		3227
				10 AE E9 0012C 19\$:	BLBC FOUND, 20\$		3234
		14	AE	0C AC D0 00130	MOVL TOKEN, TEMP_TOKEN		3236
		1C	AE	01 D0 00135	MOVL #1, CONTINUE		3237
				C4 11 00139	BRB 16\$		3234
				14 AE D5 0013B 20\$:	TSTL TEMP_TOKEN		3239
				BF 13 0013E	BEQL 16\$		
			0C AC	D0 00140	MOVL TEMP_TOKEN, TOKEN		3241
		10	AE	01 D0 00145	MOVL #1, FOUND		3242
			09	10 AE E9 00149	BLBC FOUND, 21\$		3247
18	AE	0C	BC	14 EF 0014D	EXTZV #20, #1, @TOKEN, NEGATED		3248
			01	07 11 00154	BRB 22\$		
			04	52 E8 00156 21\$:	BLBS EXPLICIT, 22\$		3249
		10	AC	01 D0 00159	MOVL #1, DEFAULT		3250
			0E	10 AC E9 0015D 22\$:	BLBC DEFAULT, 25\$		3258
		05	A9	02 E0 00161 23\$:	BBS #2, 4(PLOCK), 24\$		3259
				10 AE D4 00166	CLRL FOUND		3260
				04 11 00169	BRB 25\$		
			10	01 D0 0016B 24\$:	MOVL #1, FOUND		3261
			36	57 E9 0016F 25\$:	BLBC GET VALUE, 30\$		3267
			32	10 AE E9 00172	BLBC FOUND, 30\$		
			50	FF A6 9E 00176	MOVAB -1(R6), R0		3270
			50	04 C4 0017A	MULL2 #4, R0		
			09	10 AC E8 0017D	BLBS DEFAULT, 26\$		
				6048 9F 00181	PUSHAB (R0)[R8]		
			9E	0C AC D0 00184	MOVL TOKEN, @ (SP)+		
				19 11 00188	BRB 29\$		
				08 BE46 D5 0018A 26\$:	TSTL @8(SP)[CTX]		3271
				07 12 0018E	BNEQ 27\$		
				6048 9F 00190	PUSHAB (R0)[R8]		3272
				9E D4 00193	CLRL @ (SP)+		
				0C 11 00195	BRB 29\$		
			50	58 C0 00197 27\$:	ADDL2 R8, R0		3273
				05 14 0019A	BGTR 28\$		
			60	01 CE 0019C	MNEGL #1, (R0)		3274

		02	11	0019F	BRB	29\$		
		60	D4	001A1	CLRL	(R0)		3275
08	BE46	59	D0	001A3	MOVL	BLOCK, @8(SP)[CTX]		3276
	07	6C	91	001A8	CMPB	(AP), #7		3283
		17	1F	001AB	BLSSU	31\$		
		1C	AC	D5	TSTL	28(AP)		
		12	13	001B0	BEQL	31\$		
	01	56	D1	001B2	CMPL	CTX, #1		
		0D	12	001B5	BNEQ	31\$		
	09	10	AE	E9	BLBC	FOUND, 31\$		
	05	10	AC	E8	BLBS	DEFAULT, 31\$		
1C	BC	0C	AC	D0	MOVL	TOKEN, @QUAL		3284
		56	D6	001C4	INCL	CTX		3286
		FE	75	31	BRW	3\$		3169
	33	10	AE	E9	BLBC	FOUND, 37\$		3292
	06		6C	91	CMPB	(AP), #6		3298
		05	1F	001D0	BLSSU	33\$		
		18	AC	D5	TSTL	24(AP)		
		21	12	001D5	BNEQ	36\$		
	08	10	AC	E9	BLBC	DEFAULT, 34\$		3299
	50	00000000G	8F	D0	MOVL	#CLIS_DEFAULTED, R0		3301
			04	001E2	RET			
08	0C	BC	14	E1	BBC	#20, @TOKEN, 35\$		
		50	8F	D0	MOVL	#CLIS_NEGATED, R0		3302
			04	001EF	RET			
		50	8F	D0	MOVL	#CLIS_PRESENT, R0		3303
			04	001F7	RET			3301
		1C	10	AC	E9	BLBC	DEFAULT, 39\$	3308
	08	18	AE	E9	BLBC	NEGATED, 38\$		3309
	50	00000000G	8F	D0	MOVL	#CLIS_ABSENT, R0		3310
			04	00207	RET			
		18	AC	DD	PUSHL	RETDESC		3311
		FF	A6	9F	PUSHAB	-1(CTX)		
			59	DD	PUSHL	BLOCK		
00000000V	EF		03	FB	CALLS	#3, GET_DEFAULT_VALUE		
			04	00217	RET			3312
		18	AC	DD	PUSHL	RETDESC		
		FF	A6	9F	PUSHAB	-1(CTX)		
			59	DD	PUSHL	BLOCK		
		0C	AC	DD	PUSHL	TOKEN		
00000000V	EF		04	FB	CALLS	#4, GET_NEXT_VALUE		
			04	0022A	RET			3314

; Routine Size: 555 bytes, Routine Base: DCL\$ZCODE + 0928

```

1730 3315 1 ROUTINE get_param_token (index, retoken) =
1731 3316 1
1732 3317 1 ---
1733 3318 1
1734 3319 1 Get the next token in the command line which is a parameter value.
1735 3320 1
1736 3321 1 Inputs:
1737 3322 1
1738 3323 1 index = Address of longword containing previous token index.
1739 3324 1 retoken = Address of longword to receive token descriptor address
1740 3325 1
1741 3326 1 Outputs:
1742 3327 1
1743 3328 1 index = Address of longword containing token index of parameter.
1744 3329 1 retoken = Address of longword containing token descriptor address.
1745 3330 1
1746 3331 1 routine value = True if parameter value found, false if eol detected.
1747 3332 1 ---
1748 3333 1
1749 3334 2 BEGIN
1750 3335 2
1751 3336 2 BIND
1752 3337 2 wrk = ctl$gl_dclprstown : REF BBLOCK;
1753 3338 2
1754 3339 2 LOCAL
1755 3340 2 token: REF BBLOCK; ! Address of token descriptor
1756 3341 2
1757 3342 2 token = token_desc(..index); ! Get starting token descriptor address
1758 3343 2
1759 3344 3 WHILE (.token [ptr_v_type] NEQ ptr_k_endline) ! Get each token on the line
1760 3345 3 DO BEGIN ! Until the end of the line is reached
1761 3346 3 token = .token + ptr_c_length; ! Skip to the next token
1762 3347 3 .index = ..index + 1; ! Increment the token index
1763 3348 3
1764 3349 4 IF (.token [ptr_v_type] EQL ptr_k_parametr) ! If a parameter value was found
1765 3350 4 AND (.token [ptr_b_level] EQL 1) ! and it is at level one
1766 3351 4 THEN BEGIN ! then return success
1767 3352 4 .retoken = .token; ! Return token
1768 3353 4 RETURN true; ! and indicate found
1769 3354 3 END;
1770 3355 3
1771 3356 2 END;
1772 3357 2
1773 3358 2 RETURN false; ! Indicate no parameter found
1774 3359 2
1775 3360 1 END;

```

```

                                0000 00000 GET_PARAM_TOKEN:
                                .WORD Save nothing
                                51      04 50 00000000G 00 D0 00002      MOVL WRK, R0
                                BC      0C C5 00009      MULL3 #12, @INDEX, R1
                                50      F9AA C140 9E 0000E      MOVAB -1622(R1)(R0), TOKEN
                                04      04      1C ED 00014 1$:      CMPZV #28, #4, (TOKEN), #4
04                                : 3315
                                : 3342
                                :
                                : 3344

```

RPCLINT
V04-000

K 16
16-Sep-1984 00:26:36
14-Sep-1984 12:15:33

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCL.SRC]RPCLINT.B32;1
Page 58
(20)

				1B	13	00019	BEQL	2\$	:	
		50		0C	C0	0001B	ADDL2	#12, TOKEN	:	3346
			04	BC	D6	0001E	INCL	@INDEX	:	3347
03				1C	ED	00021	CMPZV	#28, #4, (TOKEN), #3	:	3349
	60		04	EC	12	00026	BNEQ	1\$	:	
		01		A0	91	00028	CMPB	4(TOKEN), #1	:	3350
			04	E6	12	0002C	BNEQ	1\$	:	
	08	BC		50	D0	0002E	MOVL	TOKEN, @RETTOKEN	:	3352
		50		01	D0	00032	MOVL	#1, R0	:	3353
					04	00035	RET		:	
				50	D4	00036	CLRL	R0	:	3358
				04	00038	2\$: RET			:	3360

; Routine Size: 57 bytes, Routine Base: DCL\$ZCODE + 0B53

```

1777 3361 1 ROUTINE get_next_value (token, entity, ctx, retdesc) =
1778 3362 1 ---
1779 3363 1
1780 3364 1     Get the next explicit or default value in the current value list.
1781 3365 1
1782 3366 1 Inputs:
1783 3367 1
1784 3368 1     token = address of the last token examined
1785 3369 1     entity = address of the last entity descriptor block examined
1786 3370 1     ctx = current context level
1787 3371 1     retdesc = address of return value descriptor
1788 3372 1
1789 3373 1 Outputs:
1790 3374 1
1791 3375 1     retdesc is filled in
1792 3376 1     routine value is true if value is found, else false
1793 3377 1
1794 3378 1 ---
1795 3379 1
1796 3380 2 BEGIN
1797 3381 2
1798 3382 2 MAP
1799 3383 2     token : REF BBLOCK;
1800 3384 2
1801 3385 2 BIND
1802 3386 2     entity_context = ctl$gl_clintown [dcl_l_entity] : VECTOR, ! Entity context array
1803 3387 2     token_context = ctl$gl_clintown [dcl_l_token] : VECTOR; ! Token context array
1804 3388 2
1805 3389 2 LOCAL
1806 3390 2     found; ! Value found flag
1807 3391 2
1808 3392 2 IF .token_context [.ctx] EQL -1 ! Have we already exhausted these values?
1809 3393 2 THEN RETURN cli$_absent; ! Yes, then return null string
1810 3394 2
1811 3395 3 IF (.token_context [.ctx] GTR 0) ! If there is an extant value context
1812 3396 3 AND (.entity_context [.ctx + 1] GTR 0) ! and if we are backing up a level
1813 3397 2 THEN token_context [.ctx] = 0; ! Then get the first value
1814 3398 2
1815 3399 2 IF .token_context [.ctx] EQL 0 ! Is there an extant value context?
1816 3400 3 THEN IF (found = get_explicit_value (token, 1)) ! No, then find the first value
1817 3401 2 THEN token_context [.ctx] = .token ! and save it's location as context
1818 3402 2 ELSE IF .token [ptr_v_negate] ! Not found, was the previous keyword negate
1819 3403 2 THEN RETURN cli$_absent ! Yes, then return absent
1820 3404 2 ELSE RETURN get_default_value ! No, return default value
1821 3405 2     (.entity, .ctx, .retdesc);
1822 3406 2
1823 3407 2 token = .token_context [.ctx]; ! Set the value context
1824 3408 2 get_specified_value (.token, .retdesc); ! Get the qualifier value
1825 3409 2
1826 3410 2 IF found = get_explicit_value (token, 0) ! Is there a next value?
1827 3411 2 THEN token_context [.ctx] = .token ! Yes, then set the new context
1828 3412 3 ELSE BEGIN ! No,
1829 3413 3     token_context [.ctx] = -1; ! Then invalidate the context
1830 3414 3     found = true; ! But return that the current value was found
1831 3415 2 END;
1832 3416 2
1833 3417 2 zero_context_arrays (.ctx + 1); ! Zero the context past this point

```

: 1834
: 1835
: 1836

3418 2
3419 2 RETURN .found;
3420 1 END;

! Return generic status

```

                                OFFC 00000 GET_NEXT_VALUE:
                                .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
5B 00000000V EF 9E 00002 MOVAB GET_EXPLICIT_VALUE, R11
50 00000000G 00 D0 00009 MOVL CTL$GL_C1INTOWN, R0
53 40 A0 9E 00010 MOVAB 64(R0), R3
59 5C A0 9E 00014 MOVAB 92(R0), R9
56 0C AC D0 00018 MOVL CTX, R6
52 6946 DE 0001C MOVAL (R9)[R6], R2
8F FFFFFFFF 62 D1 00020 CMPL (R2), #-1
29 13 00027 BEQL 3$
62 D5 00029 TSTL (R2)
08 15 0002B BLEQ 1$
04 A346 D5 0002D TSTL 4(R3)[R6]
02 15 00031 BLEQ 1$
62 D4 00033 CLRL (R2)
62 D5 00035 1$: TSTL (R2)
31 12 00037 BNEQ 5$
01 DD 00039 PUSHL #1
04 AC 9F 0003B PUSHAB TOKEN
68 02 FB 0003E CALLS #2, GET_EXPLICIT_VALUE
58 50 D0 00041 MOVL R0, FOUND
06 58 E9 00044 BLBC FOUND, 2$
62 04 AC D0 00047 MOVL TOKEN, (R2)
1D 11 0004B BRB 5$
08 04 BC 14 E1 0004D 2$: BBC #2C ,TOKEN, 4$
50 00000000G 8F D0 00052 3$: MOVL #CL _ABSENT, R0
04 00059 RET
10 AC DD 0005A 4$: PUSHL RETDESC
56 DD 0005D PUSHL R6
08 AC DD 0005F PUSHL ENTITY
03 FB 00062 CALLS #3, GET_DEFAULT_VALUE
04 00069 RET
04 AC 62 D0 0006A 5$: MOVL (R2), TOKEN
10 AC DD 0006E PUSHL RETDESC
04 AC DD 00071 PUSHL TOKEN
00000000V EF 02 FB 00074 CALLS #2, GET_SPECIFIED_VALUE
7E D4 0007B CLRL -(SP)
04 AC 9F 0007D PUSHAB TOKEN
68 02 FB 00080 CALLS #2, GET_EXPLICIT_VALUE
58 50 D0 00083 MOVL R0, FOUND
06 58 E9 00086 BLBC FOUND, 6$
62 04 AC D0 00089 MOVL TOKEN, (R2)
06 11 0008D BRB 7$
62 01 CE 0008F 6$: MNEGL #1, (R2)
58 01 D0 00092 MOVL #1, FOUND
50 FA A6 9E 00095 7$: MOVAB -6(R6), R0
50 04 C4 00099 MULL2 #4, R0
57 50 CE 0009C MNEGL R0, R7
5A 04 A346 DE 0009F MOVAL 4(R3)[R6], R10

```


RPCLINT
V04-000

B 1
16-Sep-1984 00:26:36
14-Sep-1984 12:15:33

VAX-11 Bliss-32 V4.0-742
DISK\$VMMASTER:[DCL.SRC]RPCLINT.B32;1 Page 61
(21)

57	00	6E	00	2C	000A4	MOVCS	#0, (SP), #0, R7, (R10)	:
			6A		000A9			:
		5A	04	A946	DE	000AA	MOVAL	4(R9)[R6], R10
57	00	6E	00	2C	000AF	MOVCS	#0, (SP), #0, R7, (R10)	:
			6A		000B4			:
		50	58	D0	000B5	MOVL	FOUND, R0	: 3419
			04	000B8	RET			: 3420

; Routine Size: 185 bytes, Routine Base: DCL\$ZCODE + 0B8C

```

1838 3421 1 ROUTINE get_explicit_value (token, level) =
1839 3422 1 ---
1840 3423 1
1841 3424 1     Get the next explicit value in the current value list.
1842 3425 1
1843 3426 1 Inputs:
1844 3427 1
1845 3428 1     token = address of address of the last token examined
1846 3429 1     level = flag, if present, get first value at the next level
1847 3430 1
1848 3431 1 Outputs:
1849 3432 1
1850 3433 1     token and level are updated
1851 3434 1     routine value = True if found, else false
1852 3435 1 ---
1853 3436 1
1854 3437 2 BEGIN
1855 3438 2
1856 3439 2 LOCAL
1857 3440 2     ptr : REF BBLOCK;
1858 3441 2
1859 3442 2
1860 3443 2     If starting a new value level, then set the level value and check that
1861 3444 2     the previous terminator is and equal sign (KEYWORD=).
1862 3445 2
1863 3446 2 ptr = ..token;
1864 3447 2 IF .level EQL 0
1865 3448 2     THEN level = .ptr [ptr_b_level]
1866 3449 2     ELSE BEGIN
1867 3450 2         IF .ptr [ptr_v_term] NEQ ptr_k_colon
1868 3451 2             THEN RETURN cli$_absent;
1869 3452 2         level = .ptr [ptr_b_level] + 1;
1870 3453 2     END;
1871 3454 2
1872 3455 2
1873 3456 2     Get the next value in the list.
1874 3457 2
1875 3458 2 WHILE (.ptr [ptr_v_type] NEQ ptr_k_endline)
1876 3459 2 DO BEGIN
1877 3460 2     ptr = .ptr + ptr_c_length;
1878 3461 2
1879 3462 2     IF .ptr [ptr_b_level] LSSU .level
1880 3463 2     THEN RETURN cli$_absent;
1881 3464 2
1882 3465 2     IF .ptr [ptr_b_level] EQL .level
1883 3466 2     THEN IF (.level NEQ 1) OR
1884 3467 2         (.ptr [ptr_v_type] EQL ptr_k_parametr)
1885 3468 2     THEN EXIT LOOP;
1886 3469 2
1887 3470 2 END;
1888 3471 2
1889 3472 2
1890 3473 2     if end of line, then return not found.
1891 3474 2
1892 3475 2 IF .ptr [ptr_v_type] EQL ptr_k_endline
1893 3476 2 THEN RETURN cli$_absent;
1894 3477 2

```

```

: Get address of last token examined
: If next value is at current level
: Then get that level from the token desc
: Else
: Verify that previous token ends with a colon
: Return no more values if not
: Indiate that we want a more deeply nested value

```

```

: While there are more tokens left to examine
: Scan for the next value
: Get the next token

```

```

: If it is shallower than the value we want
: Then return not found

```

```

: If it is the level we want
: Then if not mistaking a qualifier for a
: parameter value
: Then exit the loop

```

```

: If EOL
: Then return not found

```



```

1895 3478 2 |
1896 3479 2 | If we've gotten this far, then we've found a value. Return it.
1897 3480 2 | Also, set the success status according to the type of terminator that
1898 3481 2 | immediately preceds the value.
1899 3482 2 |
1900 3483 2 | .token = .ptr;
1901 3484 2 | ptr = .ptr - ptr_c_length;
1902 3485 2 | IF .ptr [ptr_v_term] EQL ptr_k_plus
1903 3486 2 | THEN RETURN-clis_concat
1904 3487 2 | ELSE IF .ptr [ptr_v_term] EQL ptr_k_comma
1905 3488 2 | THEN RETURN-clis_comma
1906 3489 2 | ELSE RETURN true;
1907 3490 2 |
1908 3491 1 | END;

```

```

Return pointer to token desc
Back up one descriptor
If previous value,
then return plus or comma
depending on the terminator
Return no previous value

```

				0000 00000 GET_EXPLICIT VALUE:						
			50	04	BC	D0	00002	.WORD	Save nothing	3421
				08	AC	D5	00006	MOVL	@TOKEN, PTR	3446
						07	12	TSTL	LEVEL	3447
			08	AC	04	A0	9A	BNEQ	1\$	
						10	11	MOVZBL	4(PTR), LEVEL	3448
02	03	A0				38	12	BRB	2\$	
			04			00	ED	CMPZV	#0, #4, 3(PTR), #2	3450
						38	12	BNEQ	4\$	
			08	AC	04	A0	9A	MOVZCL	4(PTR), LEVEL	3452
						AC	D6	INCL	LEVEL	
04		60				1C	ED	CMPZV	#28, #4, (PTR), #4	3458
			04			22	13	BEQL	3\$	
			50			0C	C0	ADDL2	#12, PTR	3460
08	AC	04	A0			00	ED	CMPZV	#0, #8, 4(PTR), LEVEL	3462
						1D	1F	BLSSU	4\$	
08	AC	04	A0			00	ED	CMPZV	#0, #8, 4(PTR), LEVEL	3465
			08			E4	12	BNEQ	2\$	
			01	08	AC	D1	0003E	CML	LEVEL, #1	3466
						07	12	BNEQ	3\$	
03		60				1C	ED	CMPZV	#28, #4, (PTR), #3	3467
			04			D7	12	BNEQ	2\$	
04		60				1C	ED	CMPZV	#28, #4, (PTR), #4	3475
						08	12	BNEQ	5\$	
			50	00000000G	8F	D0	00052	MOVL	#CLIS_ABSENT, R0	3476
						04	00059	RET		
			04	BC	50	D0	0005A	MOVL	PTR, @TOKEN	3483
						0C	C2	SUBL2	#12, PTR	3484
04	03	A0				00	ED	CMPZV	#0, #4, 3(PTR), #4	3485
						08	12	BNEQ	6\$	
			50	00000000G	8F	D0	00069	MOVL	#CLIS_CONCAT, R0	3487
						04	00070	RET		
05	03	A0				00	ED	CMPZV	#0, #4, 3(PTR), #5	
						08	12	BNEQ	7\$	
			50	00000000G	8F	D0	00079	MOVL	#CLIS_COMMA, R0	3488
						04	00080	RET		
			50		01	D0	00081	MOVL	#1, R0	3489
						04	00084	RET		3491

RPCLINT
V04-000

E 1
16-Sep-1984 00:26:36
14-Sep-1984 12:15:33

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCL.SRC]RPCLINT.B32;1 Page 64 (22)

; Routine Size: 133 bytes, Routine Base: DCL\$ZCODE + 0C45

```

1910 3492 1 ROUTINE get_specified_value (token, retdesc) =
1911 3493 1 ---
1912 3494 1
1913 3495 1     Get the value (possibly extending down several levels) that begins
1914 3496 1     with the specified token.
1915 3497 1
1916 3498 1     Inputs:
1917 3499 1
1918 3500 1     token = Address of the first token in the value
1919 3501 1     retdesc = Address of the descriptor to return the result in
1920 3502 1
1921 3503 1     Outputs:
1922 3504 1
1923 3505 1     retdesc is updated
1924 3506 1     routine value = always true
1925 3507 1 ---
1926 3508 1
1927 3509 1 BEGIN
1928 3510 1
1929 3511 1 MAP
1930 3512 1     retdesc : REF BBLOCK,
1931 3513 1     token : REF BBLOCK;
1932 3514 1
1933 3515 1 BIND
1934 3516 1     wrk = cti$gl_dclprstown : REF BBLOCK;
1935 3517 1
1936 3518 1 LOCAL
1937 3519 1     count,                                ! Number of tokens in value
1938 3520 1     parens,                                ! Parenthesis count
1939 3521 1     ptr : REF BBLOCK;                    ! Pointer to token after the value
1940 3522 1
1941 3523 1
1942 3524 1     Initialize the local variables.
1943 3525 1
1944 3526 1     parens = 0;                            ! Set no parenthesis seen
1945 3527 1     count = 1;                            ! Start with one token
1946 3528 1     ptr = .token + ptr_c_length;          ! Point to second token
1947 3529 1
1948 3530 1
1949 3531 1     Get all value tokens in the command that are part of this value.
1950 3532 1
1951 3533 1 WHILE ((.ptr [ptr_v_type] NEQ ptr_k_endline) AND ! While there are still tokens on the line
1952 3534 1         (.ptr [ptr_b_level] GTR .token [ptr_b_level])) ! and they are part of the current value
1953 3535 1 DO BEGIN ! Update the local variables
1954 3536 1     LOCAL index;
1955 3537 1
1956 3538 1
1957 3539 1     If token is preceeded by a "(", then increment the paren count.
1958 3540 1
1959 3541 1     IF (CH$RCHAR (.ptr [ptr_v_offset] + wrk [wrk_g_buffer] - 1) EQL %C '('
1960 3542 1     THEN parens = .parens + 1;
1961 3543 1
1962 3544 1
1963 3545 1     If token is terminated by ")"'s, then decrement the paren count
1964 3546 1     appropriately.
1965 3547 1     index = 0;
1966 3548 1     WHILE (CH$RCHAR (.ptr [ptr_v_offset] + wrk [wrk_g_buffer]

```

```

1967 3549 4
1968 3550 DO index = .index + 1;
1969 3551 parens = .parens - .index;
1970 3552
1971 3553
1972 3554 Update the last token pointer and the token count.
1973 3555
1974 3556 ptr = .ptr + ptr_c_length;
1975 3557 count = .count + 1;
1976 3558 END;
1977 3559
1978 3560
1979 3561 Strip off the terminator if appropriate and return the value that we found.
1980 3562
1981 3563 retdesc [dsc$a_pointer] = .token [ptr_v_offset] + wrk [wrk_g_buffer];
1982 3564 IF .count EQL 1
1983 3565 THEN retdesc [dsc$w_length] = .token [ptr_b_value]
1984 3566 ELSE BEGIN
1985 3567 .retdesc [dsc$w_length] = .ptr [ptr_v_offset] - .token [ptr_v_offset];
1986 3568 IF .ptr [ptr_v_type] NEQ ptr_k_endline
1987 3569 THEN retdesc [dsc$w_length] = .retdesc [dsc$w_length] - 1;
1988 3570 retdesc [dsc$w_length] = .retdesc [dsc$w_length] + .parens;
1989 3571 END;
1990 3572
1991 3573 RETURN true;
1992 3574 1 END;

```

				00FC 00000 GET_SPECIFIED_VALUE:					
				56	D4	00002	WORD	Save R2,R3,R4,R5,R6,R7	3492
				01	D0	00004	CLRL	PARENS	3526
		57		04	AC	00007	MOVL	#1, COUNT	3527
		54	04	0C	A4	9E 0000B	MOVL	TOKEN, R4	3528
04		50		00	8F	C3 0000F	MOVAB	12(R4), PTR	
	55	00000000G	00	00000B6E	1C	ED 0001B	SUBL3	#2926, WRK, R5	3541
	60		04		35	13 00020	CMPZV	#28, #4, (PTR), #4	3533
			04	A4	04	A0 91 00022	BEQL	5\$	
						2E 1B 00027	CMPB	4(PTR), 4(R4)	3534
51	01	A0	0C		00	EF 00029	9LEQU	5\$	
		52	51		55	C1 0002F	EXTZV	#0, #12, 1(PTR), R1	3541
			28	FF	A2	91 00033	ADDL3	R5, R1, R2	
					02	12 00037	CMPB	-1(R2), #40	
					56	D6 00039	BNEQ	2\$	
					53	D4 0003B	INCL	PARENS	3542
		51			60	9A 0003D	CLRL	INDEX	3547
		51			52	C0 00040	MOVZBL	(PTR), R1	3549
		29			6341	91 00043	ADDL2	R2, R1	
					04	12 00047	CMPB	(INDEX)[R1], #41	
					53	D6 00049	BNEQ	4\$	
					F0	11 0004B	INCL	INDEX	3550
		56			53	C2 0004D	BRB	3\$	
		50			0C	C0 00050	SUBL2	INDEX, PARENS	3551
					57	D6 00053	ADDL2	#12, PTR	3556
							INCL	COUNT	3557

RPCLINT
V04-000

H 1
16-Sep-1984 00:26:36
14-Sep-1984 12:15:33

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCL.SRC]RPCLINT.B32;1 Page 67
(23)

52	01	A4	51	08	C4	11	00055	BRB	1\$	3533	
	04	A1	0C		AC	D0	00057	5\$:	MOVL	RETDESC, R1	3563
			52		00	EF	0005B		EXTZV	#0, #12, 1(R4), R2	
			01		55	C1	00061		ADDL3	R5, R2, 4(R1)	
					57	D1	00066		CMPL	COUNT, #1	3564
			61		05	12	00069		BNEQ	6\$	
					64	9B	0006B		MOVZBW	(R4), (R1)	3565
					1C	11	0006E		BRB	8\$	
52	01	A0	0C		00	EF	00070	6\$:	EXTZV	#0, #12, 1(PTR), R2	3567
53	01	A4	0C		00	EF	00076		EXTZV	#0, #12, 1(R4), R3	
		61	52		53	A3	0007C		SUBW3	R3, R2, (R1)	
04		60	04		1C	ED	00080		CMPZV	#28, #4, (PTR), #4	3568
					02	13	00085		BEQL	7\$	
			61		61	B7	00087		DECW	(R1)	3569
			50		56	A0	00089	7\$:	ADDW2	PARENS, (R1)	3570
					01	D0	0008C	8\$:	MOVL	#1, R0	3573
					04	0008F			RET		3574

; Routine Size: 144 bytes, Routine Base: DCL\$ZCODE + OCCA

```

1994 3575 1 ROUTINE get_default_value (entity, ctx, retdesc) =
1995 3576 1 ---
1996 3577 1
1997 3578 1     Get the default value associated with the specified entity.
1998 3579 1
1999 3580 1     Inputs:
2000 3581 1
2001 3582 1     entity = Address of an entity descriptor block
2002 3583 1     ctx = Context level of last entity
2003 3584 1     retdesc = Address of a string descriptor to return the result in
2004 3585 1
2005 3586 1     Outputs:
2006 3587 1
2007 3588 1     retdesc is returned as described above
2008 3589 1 ---
2009 3590 1
2010 3591 2 BEGIN
2011 3592 2
2012 3593 2 MAP
2013 3594 2     entity : REF BBLOCK,
2014 3595 2     retdesc : REF BBLOCK;
2015 3596 2
2016 3597 2 BIND
2017 3598 2     wrk = ctl$gl_dclprstown : REF BBLOCK,           ! Address of command work area
2018 3599 2     entity_context = ctl$gl_clintown [dcl_l_entity] : VECTOR, ! Entity context array
2019 3600 2     token_context = ctl$gl_clintown [dcl_l_token] : VECTOR;   ! Token context array
2020 3601 2
2021 3602 2 LOCAL
2022 3603 2     found,                                           ! Value found flag
2023 3604 2     string : BBLOCK [dsc$e_s_bln],                 ! Local descriptor for value
2024 3605 2     value : REF VECTOR [,BYTE];                     ! Address of ASCII value
2025 3606 2
2026 3607 2
2027 3608 2     Initialize the default value buffer
2028 3609 2
2029 3610 2     ctl$gl_clintown [dcl_w_deflen] = 0;               ! Clear default value buffer
2030 3611 2
2031 3612 2
2032 3613 2     If there is a default value associated with the entity, and we have not
2033 3614 2     returned it before, then return it now.
2034 3615 2
2035 3616 2     IF .entity [ent_w_defval] NEQ 0                     ! If default value
2036 3617 2     THEN IF (.entity_context [.ctx] EQL .entity)       ! Then, if returned before
2037 3618 2         AND (.token_context [.ctx] EQL -1)
2038 3619 2         THEN RETURN cli$_absent                         ! Then return not found
2039 3620 2     ELSE BEGIN                                           ! Else return the value
2040 3621 2         value = .entity + .entity [ent_w_defval] + 1;   ! Get address of ASCII string
2041 3622 2         retdesc [dsc$w_length] = .value [0];           ! Get default value
2042 3623 2         retdesc [dsc$a_pointer] = value [1];
2043 3624 2         token_context [.ctx] = -1;
2044 3625 2         RETURN true;
2045 3626 2     END;
2046 3627 2
2047 3628 2
2048 3629 2     If there is no keyword list associated with the entity. Then it can no
2049 3630 2     longer have a default value.
2050 3631 2

```



```

2051 3632 2 IF .entity [ent_l_user_type] EQL 0          | If no keyword list
2052 3633      THEN RETURN cli$absent;              | Return no value
2053 3634
2054 3635      |
2055 3636      | If we have a previous keyword context, then use it.
2056 3637
2057 3638      ctx = .ctx + 1;
2058 3639      IF (.entity_context [ctx] GTR 0)
2059 3640          AND (.entity_context [ctx + 1] EQL 0)
2060 3641      THEN BEGIN
2061 3642          entity = .entity_context [ctx];
2062 3643          IF .entity [ent_l_next] EQL 0
2063 3644              THEN RETURN cli$absent
2064 3645              ELSE entity = .entity [ent_l_next]
2065 3646                  + .wrk [wrk_l_tab_vec];
2066 3647      END
2067 3648      ELSE BEGIN
2068 3649          entity = .entity [ent_l_user_type] + .wrk [wrk_l_tab_vec];
2069 3650          entity = .entity [ent_l_next] + .wrk [wrk_l_tab_vec];
2070 3651      END;
2071 3652      zero_context_arrays (.ctx + 1);
2072 3653
2073 3654      |
2074 3655      | Find the next keyword that is present by default. Return it and any
2075 3656      | default value that may be associated with it.
2076 3657
2077 3658      found = cli$absent;
2078 3659      WHILE (.entity NEQ 0)
2079 3660      DO BEGIN
2080 3661          IF .entity [ent_v_deftrue]
2081 3662              THEN BEGIN
2082 3663              |
2083 3664              | If .found
2084 3665              | THEN RETURN cli$comma
2085 3666              | ELSE found = true;
2086 3667              |
2087 3668              value = .entity + .entity [ent_w_name];
2088 3669              token_context [ctx] = 0;
2089 3670              entity_context [ctx] = .entity;
2090 3671              |
2091 3672              IF (.entity [ent_l_user_type] NEQ 0)
2092 3673                  OR (.entity [ent_w_defval] NEQ 0)
2093 3674              THEN BEGIN
2094 3675                  string [dsc$w_length] = .value [0];
2095 3676                  string [dsc$a_pointer] = value [1];
2096 3677                  insert_string (string);
2097 3678                  insert_next_level (.entity);
2098 3679                  CH$MOVE (dsc$w_s_bln,
2099 3680                      cti$gl_clintown [dcl_w_deflen], .retdesc);
2100 3681              END
2101 3682              ELSE BEGIN
2102 3683                  retdesc [dsc$w_length] = .value [0];
2103 3684                  retdesc [dsc$a_pointer] = value [1];
2104 3685              END;
2105 3686
2106 3687      END;
2107 3688

```

| If no keyword list
 | Return no value

 | If we have a previous keyword context, then use it.
 | Increment context level
 | If we have a previous context
 | but are not backing up a level
 | Then use it
 | Get last keyword returned
 | If no more keywords
 | Then return no value
 | Else point to next

 | Else start with first keyword
 | Skip list header

 | Zero the context arrays from this point

 | Find the next keyword that is present by default. Return it and any
 | default value that may be associated with it.

 | Assume no value will be found
 | Loop will be exited by EXITLOOP

 | If keyword is present by default
 | Then return it

 | If a value was already found
 | Then return "another value" status
 | Else mark value found

 | Get address of ASCII string
 | Mark entity defaulted

 | If keyword can have
 | a default value
 | Then process it
 | Get keyword name

 | Insert keyword into buffer
 | Insert its def val into buffer
 | Get the result

 | Else simply return the keyword
 | Get keyword name

```

! If no more keywords
! Then return status
! Get next keyword

```

OFFC 00000 GET_DEFAULT VALUE:											
	5B	00000000G	00	9E	00002		.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11		3575	
	5E		08	C2	00009		MOVAB	WRK, R11			
	59	00000000G	00	00	0000C		SUBL2	#8, SP			
		0084	C9	B4	00013		MOVL	CTL\$GL CLINTOWN, R9		3599	
	52		04	AC	D0	00017	CLRW	132(R9)		3610	
			1C	A2	B5	0001B	MOVL	ENTITY, R2		3616	
				33	13	0001E	TSTW	28(R2)			
	50		08	AC	D0	00020	BEQL	2\$			
	52		40	A940	D1	00024	MOVL	CTX, R0		3617	
				0B	12	00029	CMPL	64(R9)[R0], R2			
FFFFFFFF	8F		5C	A940	D1	0002B	BNEQ	1\$		3618	
				3F	13	00034	CMPL	92(R9)[R0], #-1			
	51		1C	A2	3C	00036	BEQL	3\$		3621	
	56		01	A241	9E	0003A	MOVZWL	28(R2), R1			
	51		0C	AC	D0	0003F	MOVAB	1(R2)[R1], VALUE		3622	
	61			66	9B	00043	MOVL	RETDESC, R1			
04	A1		01	A6	9E	00046	MOVZBW	(VALUE), (R1)		3623	
5C	A940			01	CE	0004B	MOVAB	1(R6), 4(R1)		3624	
				00F1	31	00050	MNEGL	#1, 92(R9)[R0]		3625	
				10	A2	D5	BRW	13\$		3632	
					1D	13	TSTL	16(R2)			
				08	AC	D6	BEQL	3\$			
	57		08	AC	D0	0005B	INCL	CTX		3638	
	50		40	A947	D0	0005F	MOVL	CTX, R7		3639	
				23	15	00064	MOVL	64(R9)[R7], R0			
			44	A947	D5	00066	BLEQ	5\$		3640	
				1D	12	0006A	TSTL	68(R9)[R7]			
04	AC			50	D0	0006C	BNEQ	5\$		3642	
			08	A0	D5	00070	MOVL	R0, ENTITY		3643	
				08	12	00073	TSTL	8(R0)			
	50	00000000G	8F	D0	00075		BNEQ	4\$		3644	
				04	0007C		MOVL	#CLIS_ABSENT, R0			
	51			6B	D0	0007D	RET			3646	
04	AC	08	A0	DE	A1	C1	MOVL	WRK, R1			
					15	11	ADDL3	-34(R1), 8(R0), ENTITY		3639	
	50			6B	D0	00089	BRB	6\$		3649	
04	AC	10	A2	DE	A0	C1	MOVL	WRK, R0			
	51			04	AC	D0	ADDL3	-34(R0), 16(R2), ENTITY		3650	
04	AC	08	A1	DE	A0	C1	MOVL	ENTITY, R1			
	50			FA	A7	9E	ADDL3	-34(R0), 8(R1), ENTITY		3652	
	50				04	C4	MOVAB	-6(R7), R0			
	58				50	CE	MULL2	#4, R0			
	5A		44	A947	DE	000A8	MNEGL	R0, R8			
							MOVAL	68(R9)[R7], R10			

58	00	6E	00	2C	000AD	MOVCS	#0, (SP), #0, R8, (R10)	:	
			6A		000B2			:	
		5A	60	A947	DE	000B3	MOVAL	96(R9)[R7], R10	:
58	00	6E		00	2C	000B8	MOVCS	#0, (SP), #0, R8, (R10)	:
				6A		000BD		:	
		5A	00000000G	8F	D0	000BE	MOVL	#CLIS_ABSENT, FOUND	3658
		58	04	AC	D0	000C5	7\$:	MOVL	ENTITY, R8
				79	13	000C9	BEQL	13\$	3659
	5F	04		A8	02	E1	000CB	BBC	#2, 4(R8), 11\$
		08		5A	E9	000D0	BLBC	FOUND, 8\$	3661
		50	00000000G	8F	D0	000D3	MOVL	#CLIS_COMMA, R0	3664
				04		000DA	RET		3665
		5A		01	D0	000DB	8\$:	MOVL	#1, FOUND
		56	16	A8	3C	000DE	MOVZWL	22(R8), VALUE	3666
		56		58	C0	000E2	ADDL2	R8, VALUE	3668
			5C	A947	D4	000E5	CLRL	92(R9)[R7]	3669
	40	A947		58	D0	000E9	MOVL	R8, 64(R9)[R7]	3670
		50	01	A6	9E	000EE	MOVAB	1(R6), R0	3676
		52	0C	AC	D0	000F2	MOVL	RETDESC, R2	3680
			10	A8	D5	000F6	TSTL	16(R8)	3672
				05	12	000F9	BNEQ	9\$	
			1C	A8	B5	000FB	TSTW	28(R8)	3673
				28	13	000FE	BEQL	10\$	
		04	6E		66	9B	00100	9\$:	MOVZBW
			AE		50	D0	00103	MOVL	(VALUE), STRING
					5E	DD	00107	PUSHL	R0, STRING+4
		00000000V	EF		01	FB	00109	CALLS	SP
					58	DD	00110	PUSHL	#1, INSERT_STRING
		00000000V	EF		01	FB	00112	CALLS	R8
					00	D0	00119	CALLS	#1, INSERT_NEXT_LEVEL
	62	0084	C0	00000000G	08	28	00120	MOVL	CTL\$GL_CLINTOWN, R0
					07	11	00126	MOVCS	#8, 132(R0), (R2)
					66	9B	00128	BRB	11\$
		04	62		50	D0	0012B	10\$:	MOVZBW
			A2		50	D0	0012B	MOVL	(VALUE), (R2)
				08	A8	D5	0012F	11\$:	R0, 4(R2)
				04	12	00132	TSTL	8(R8)	3684
			50		5A	D0	00134	BNEQ	12\$
					04		00137	MOVL	FOUND, R0
			50		6B	D0	00138	RET	
	04	AC	08	A8	DE	A0	C1	0013B	12\$:
					81	11	00142	MOVL	WRK, R0
			50		01	D0	00144	ADDL3	-34(R0), 8(R8), ENTITY
					04		00147	BRB	7\$
								13\$:	MOVL
								RET	#1, R0
									3691
									3659
									3694
									3695

; Routine Size: 328 bytes, Routine Base: DCL\$ZCODE + 0D5A

```

2116 3696 1 ROUTINE insert_next_level (entity) =
2117 3697 1 ----
2118 3698 1
2119 3699 1 Put the next level of default values associated with the specified
2120 3700 1 entity into the default value buffer.
2121 3701 1
2122 3702 1 Inputs:
2123 3703 1
2124 3704 1 entity = Address of an entity descriptor block
2125 3705 1
2126 3706 1 Outputs:
2127 3707 1
2128 3708 1 The default value buffer is updated.
2129 3709 1 ----
2130 3710 1
2131 3711 2 BEGIN
2132 3712 2
2133 3713 2 MAP
2134 3714 2 entity : REF BBLOCK;
2135 3715 2
2136 3716 2 BIND
2137 3717 2 wrk = ctl$gl_dclprsw : REF BBLOCK; ! Address of command work area
2138 3718 2
2139 3719 2 LOCAL
2140 3720 2 first, ! First value found flag
2141 3721 2 string : BBLOCK [dsc$sc_s_bln], ! Local descriptor for value
2142 3722 2 value : REF VECTOR [,BYTE]; ! Address of ASCII value
2143 3723 2
2144 3724 2
2145 3725 2 If there is a default value associated with the entity, then return it now.
2146 3726 2
2147 3727 2 IF .entity [ent_w_defval] NEQ 0 ! If default value
2148 3728 2 THEN BEGIN
2149 3729 2 insert_char (XC '=' ); ! Insert an equals sign
2150 3730 2 value = .entity + .entity [ent_w_defval] + 1; ! Get address of ASCII string
2151 3731 2 string [dsc$w_length] = .value [0]; ! Get default value
2152 3732 2 string [dsc$a_pointer] = value [1];
2153 3733 2 insert_string (string); ! Insert the default value
2154 3734 2 RETURN true; ! Return found
2155 3735 2 END;
2156 3736 2
2157 3737 2
2158 3738 2 If there is no keyword list associated with the entity. Then it can no
2159 3739 2 longer have a default value.
2160 3740 2
2161 3741 2 IF .entity [ent_l_user_type] EQL 0 ! If no keyword list
2162 3742 2 THEN RETURN false; ! Return no value
2163 3743 2
2164 3744 2
2165 3745 2 Get the address of the first keyword. Set the first value flag.
2166 3746 2
2167 3747 2 entity = .entity [ent_l_user_type] + .wrk [wrk_l_tab_vec]; ! Start with first keyword
2168 3748 2 entity = .entity [ent_l_next] + .wrk [wrk_l_tab_vec]; ! Skip list header
2169 3749 2 first = true;
2170 3750 2
2171 3751 2
2172 3752 2 Find each keyword that is present by default. Insert it and any default

```

```

2173 3753 2 ! value that may be associated with it.
2174 3754 2
2175 3755 3 WHILE (.entity NEQ 0) ! Loop will be exited by EXITLOOP
2176 3756 3 DO BEGIN
2177 3757 4 IF .entity [ent_v_deftrue] ! If keyword is present by default
2178 3758 4 THEN BEGIN ! Then insert it
2179 3759 5 IF .first ! If first value
2180 3760 5 THEN BEGIN
2181 3761 6 first = false; ! Clear the flag
2182 3762 6 insert_char (%C'='); ! Insert an equals sign
2183 3763 6 insert_char (%C'('); ! Insert an open parenthesis
2184 3764 6 END
2185 3765 6 ELSE insert_char (%C','); ! Else insert a comma
2186 3766 6 value = .entity + .entity [ent_w_name]; ! Get address of ASCII string
2187 3767 6 string [dsc$w_length] = .value-1; ! Get keyword name
2188 3768 6 string [dsc$a_pointer] = value + 1;
2189 3769 6 insert_string(string); ! Insert keyword into buffer
2190 3770 6 insert_next_level (.entity); ! Insert its def val into buffer
2191 3771 6 END;
2192 3772 3
2193 3773 3 IF .entity [ent_l_next] EQL 0 ! If no more keywords
2194 3774 3 THEN EXITLOOP; ! Then done
2195 3775 3 entity = .entity [ent_l_next] + .wrk [wrk_l_tab_vec]; ! Get next keyword
2196 3776 3 END;
2197 3777 2
2198 3778 2 IF NOT .first ! If we have an open paren
2199 3779 2 THEN insert_char (%C')'); ! Then match it
2200 3780 2
2201 3781 2 RETURN true;
2202 3782 1 END;

```

00FC 00000 INSERT_NEXT_LEVEL:

57	00000000G	00	9E	00002	WORD	Save R2,R3,R4,R5,R6,R7	3696
56	00000000V	EF	9E	00009	MOVAB	WRK, R7	
55	00000000V	EF	9E	00010	MOVAB	INSERT_STRING, R6	
5E		08	C2	00017	MOVAB	INSERT_CHAR, R5	
52	04	AC	D0	0001A	SUBL2	#8, SP	
	1C	A2	B5	0001E	MOVL	ENTITY, R2	3727
		1D	13	00021	TSTW	28(R2)	
		3D	DD	00023	BEQL	1\$	
65		01	FB	00025	PUSHL	#61	3729
50	1C	A2	3C	00028	CALLS	#1, INSERT_CHAR	
54	01	A240	9E	0002C	MOVZWL	28(R2), R0	3730
6E		64	9B	00031	MOVAB	1(R2)[R0], VALUE	
04	AE	01	A4	9E	MOVZBW	(VALUE), STRING	3731
		5E	DD	00039	MOVAB	1(R4), STRING+4	3732
66		01	FB	0003B	PUSHL	SP	3733
		6F	11	0003E	CALLS	#1, INSERT_STRING	
		10	A2	D5	BRB	7\$	3734
		6E	13	00043	TSTL	16(R2)	3741
		67	D0	00045	BEQL	8\$	
04	AC	10	A2	DE	MOVL	WRK, R0	3747
		A0	C1	00048	ADDL3	-34(R0), 16(R2), ENTITY	

04	AC	08	51	04	AC	D0	0004F	MOVL	ENTITY, R1	:	3748
			A1	DE	A0	C1	00053	ADDL3	-34(R0), 8(R1), ENTITY	:	
			53		01	D0	0005A	MOVL	#1, FIRST	:	3749
			52	04	AC	D0	0005D	MOVL	ENTITY, R2	:	3755
					44	13	00061	BEQL	6\$	:	
	2E	04	A2		02	E1	00063	BBC	#2, 4(R2), 5\$	:	3757
			0B		53	E9	00068	BLBC	FIRST, 3\$	:	3759
					53	D4	0006B	CLRL	FIRST	:	3761
					3D	DD	0006D	PUSHL	#61	:	3762
			65		01	FB	0006F	CALLS	#1, INSERT_CHAR	:	
					28	DD	00072	PUSHL	#40	:	3763
					02	11	00074	BRB	4\$	:	
					2C	DD	00076	PUSHL	#44	:	3765
			65		01	FB	00078	CALLS	#1, INSERT_CHAR	:	
			54	16	A2	3C	0007B	MOVZWL	22(R2), VALUE	:	3766
			54		52	C0	0007F	ADDL2	R2, VALUE	:	
			6E		64	9B	00082	MOVZBW	(VALUE), STRING	:	3767
		04	AE	01	A4	9E	00085	MOVAB	1(R4), STRING+4	:	3768
					5E	DD	0008A	PUSHL	SP	:	3769
			66		01	FB	0008C	CALLS	#1, INSERT_STRING	:	
					52	DD	0008F	PUSHL	R2	:	3770
		FF6A	CF		01	FB	00091	CALLS	#1, INSERT_NEXT_LEVEL	:	
				08	A2	D5	00096	TSTL	8(R2)	:	3773
					0C	13	00099	BEQL	6\$	:	
			50		67	D0	0009B	MOVL	WRK, R0	:	3775
04	AC	08	A2	DE	A0	C1	0009E	ADDL3	-34(R0), 8(R2), ENTITY	:	
					B6	11	000A5	BRB	2\$	:	3755
			05		53	E8	000A7	BLBS	FIRST, 7\$	:	3778
					29	DD	000AA	PUSHL	#41	:	3779
			65		01	FB	000AC	CALLS	#1, INSERT_CHAR	:	
			50		01	D0	000AF	MOVL	#1, R0	:	3781
						04	000B2	RET		:	
					50	D4	000B3	CLRL	R0	:	3782
					04	000B5	RET			:	

; Routine Size: 182 bytes, Routine Base: DCL\$ZCODE + 0EA2

```

2204 3783 1 ROUTINE insert_string (string) =
2205 3784 1
2206 3785 1 ---
2207 3786 1
2208 3787 1     Insert the specified string into the default value buffer.
2209 3788 1
2210 3789 1 Inputs:
2211 3790 1
2212 3791 1     string = Address of the string descriptor of the value to insert
2213 3792 1
2214 3793 1 Outputs:
2215 3794 1
2216 3795 1     the default value buffer is modified as described above
2217 3796 1 ---
2218 3797 1
2219 3798 2 BEGIN
2220 3799 2
2221 3800 2 MAP
2222 3801 2     string : REF BBLOCK;
2223 3802 2
2224 3803 2 BIND
2225 3804 2     retdesc = ctl$gl_clintown [dcl_w_deflen] : BBLOCK,      ! Default value string descriptor
2226 3805 2     size = ctl$gl_clintown [dcl_w_buflen] : WORD;          ! Default value buffer size
2227 3806 2
2228 3807 2
2229 3808 2 ! If default buffer cannot fit string, then increase its size.
2230 3809 2
2231 3810 2 IF (.size - .retdesc[dsc$w_length]) LSSU .string [dsc$w_length] ! If not enough space in the buffer
2232 3811 2 THEN allocate_default_buffer (.string [dsc$w_length]);      ! Then increase its size
2233 3812 2
2234 3813 2
2235 3814 2 ! Insert the string.
2236 3815 2
2237 3816 2 CH$MOVE (.string [dsc$w_length], .string [dsc$a_pointer],      ! Insert the string
2238 3817 2     .retdesc [dsc$a_pointer] + .retdesc [dsc$w_length]);
2239 3818 2 retdesc [dsc$w_length] = .retdesc [dsc$w_length]              ! Update the length
2240 3819 2     + .string [dsc$w_length];
2241 3820 2 RETURN true;
2242 3821 2
2243 3822 1 END;

```

			00FC 00000	INSERT_STRING:		
				.WORD	Save R2,R3,R4,R5,R6,R7	
	50	00000000G	00 D0 00002	MOVL	CTL\$GL CLINTOWN, R0	3783
	57	0084	C0 9E 00009	MOVAB	132(R0), R7	3804
	50	008D	C0 3C 0000E	MOVZWL	141(R0), R0	3810
	51		67 3C 00013	MOVZWL	(R7), R1	
	50		51 C2 00016	SUBL2	R1, R0	
	56	04	AC D0 00019	MOVL	STRING, R6	
50	66	10	00 ED 0001D	CMPZV	#0, #16 (R6), R0	
			0A 1B 00022	BLEQU	1\$	
		7E	66 3C 00024	MOVZWL	(R6), -(SP)	3811
		EF	01 FB 00027	CALLS	#1, ALLOCATE_DEFAULT_BUFFER	
		00000000V				

RPCLINT
V04-000

D 2
16-Sep-1984 00:26:36
14-Sep-1984 12:15:33

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCL.SRC]RPCLINT.B32;1 Page 76
(26)

		50		67	3C	0002E	1\$:	MOVZWL	(R7), R0	:	3817
		50		A7	C0	00031		ADDL2	4(R7), R0	:	
60	04	B6		66	28	00035		MOVCL	(R6), @4(R6), (R0)	:	
		67		66	A0	C003A		ADDW2	(R6), (R7)	:	3819
		50		01	D0	0003D		MOVL	#1, R0	:	3820
				04	04	00040		RET		:	3822

: Routine Size: 65 bytes. Routine Base: DCL\$ZCODE + 0F58


```

: 2245 3823 1 ROUTINE insert_char (char) =
: 2246 3824 1
: 2247 3825 1 ---
: 2248 3826 1
: 2249 3827 1     Insert the specified character into the default value buffer.
: 2250 3828 1
: 2251 3829 1 Inputs:
: 2252 3830 1
: 2253 3831 1     char = The value of the character to insert
: 2254 3832 1
: 2255 3833 1 Outputs:
: 2256 3834 1
: 2257 3835 1     the default value buffer is modified as described above
: 2258 3836 1 ---
: 2259 3837 1
: 2260 3838 2 BEGIN
: 2261 3839 2
: 2262 3840 2 MAP
: 2263 3841 2     char : BYTE;
: 2264 3842 2
: 2265 3843 2 BIND
: 2266 3844 2     retdesc = ctl$gl_clintown [dcl_w_deflen] : BBLOCK,      ! Default value string descriptor
: 2267 3845 2     size = ctl$gl_clintown [dcl_w_bufen] : WORD;          ! Default value buffer size
: 2268 3846 2
: 2269 3847 2
: 2270 3848 2     ! If default buffer cannot fit string, then increase its size.
: 2271 3849 2
: 2272 3850 2 IF (.size - .retdesc[dsc$w_length]) LSSU 1                ! If not enough space in the buffer
: 2273 3851 2     THEN allocate_default_buffer (1);                      ! Then increase its size
: 2274 3852 2
: 2275 3853 2
: 2276 3854 2     ! Insert the character.
: 2277 3855 2
: 2278 3856 2     CH$WCHAR (.char, .retdesc [dsc$a_pointer] +             ! Insert the character
: 2279 3857 2     .retdesc [dsc$w_length]);
: 2280 3858 2     retdesc [dsc$w_length] = .retdesc [dsc$w_length] + 1;    ! Update the length
: 2281 3859 2     RETURN true;
: 2282 3860 2
: 2283 3861 1 END;

```

```

                                0004 00000 INSERT_CHAR:
                                .WORD
                                Save R2
                                CTL$GL CLINTOWN, R0
                                132(R0), R2
                                141(R0), R0
                                (R2), R1
                                R1, R0
                                SUBL2
                                BNEQ 1$
                                PUSHL #1
                                #1, ALLOCATE_DEFAULT_BUFFER
                                (R2), R0
                                4(R2), R0
                                ADDL2
                                MOVB CHAR, (R0)

```

RPCLINT
V04-000

F 2
16-Sep-1984 00:26:36
14-Sep-1984 12:15:33

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCL.SRC]RPCLINT.B32;1 Page 78 (27)

50	62	B6	0002F	INCW	(R2)	:	3858
	01	D0	00031	MOVL	#1, R0	:	3859
		04	00034	RET		:	3861

; Routine Size: 53 bytes, Routine Base: DCL\$ZCODE + 0F99


```

2285 3862 1 ROUTINE allocate_default_buffer (length) =
2286 3863 1
2287 3864 1 ---
2288 3865 1
2289 3866 1 Expand the default value buffer by at least the specified string length.
2290 3867 1
2291 3868 1 Inputs:
2292 3869 1
2293 3870 1 length = the size of the most recent string being inserted
2294 3871 1
2295 3872 1 Outputs:
2296 3873 1
2297 3874 1 --- the default value buffer is modified as described above
2298 3875 1
2299 3876 1
2300 3877 2 BEGIN
2301 3878 2
2302 3879 2 BIND
2303 3880 2 retdesc = ctl$gl_clintown [dcl_w_deflen] : BBLOCK, ! Default value string descriptor
2304 3881 2 size = ctl$gl_clintown [dcl_w_bufen] : WORD; ! Default value buffer size
2305 3882 2
2306 3883 2 LITERAL
2307 3884 2 slot = 128; ! Increments to increase the buffer size by
2308 3885 2
2309 3886 2 LOCAL
2310 3887 2 address,
2311 3888 2 old_size,
2312 3889 2 status;
2313 3890 2
2314 3891 2 old_size = .size; ! Get old size
2315 3892 2 size = ((.retdesc [dsc$w_length] + .length / slot) + 1) * slot; ! Calculate size of new buffer
2316 3893 4 IF NOT (status = (.ctl$gl_clintown [dcl_l_getvm] ! Get new buffer
2317 3894 3 (.size, address)))
2318 3895 2 THEN SIGNAL (.status); ! Signal any error
2319 3896 2 CH$MOVE (.retdesc [dsc$w_length], .retdesc [dsc$a_pointer], ! Copy old value
2320 3897 2 .address);
2321 3898 2 (.ctl$gl_clintown [dcl_l_freevm]) (old_size, ! Free old buffer
2322 3899 2 retdesc [dsc$a_pointer]);
2323 3900 2 retdesc [dsc$a_pointer] = .address; ! Save new buffer
2324 3901 2 RETURN true;
2325 3902 1 END;

```

00FC 0000 ALLOCATE_DEFAULT_BUFFER:

					WORD	Save R2,R3,R4,R5,R6,R7	3862
		57	00000000G	00	9E	00002	
		5E		08	C2	00009	
		50		67	D0	0000C	
		56	0084	C0	9E	0000F	3880
		52	008D	C0	9E	00014	
	04	AE		62	3C	00019	3881
51	04	AC	00000080	8F	C7	0001D	3891
		53		66	3C	00026	3892
		51		53	C0	00029	
						ADDL2	
						R3, R1	

RPCLINT
V04-000

H 2
16-Sep-1984 00:26:36
14-Sep-1984 12:15:33

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCL.SRC]RPCLINT.B32;1 Page 80
(28)

51	51	07	78	0002C	ASHL	#7, R1, R1	:
62	51	8F	A1	00030	ADDW3	#128, R1, (R2)	:
		8F	BB	00036	PUSHR	#^M<R2, SP>	: 3894
	7C	02	FB	0003A	CALLS	#2, @124(R0)	:
	B0	50	E8	0003E	BLBS	STATUS, 1\$	:
	09	50	DD	00041	PUSHL	STATUS	: 3895
00	00000000G	01	FB	00043	CALLS	#1, LIB\$SIGNAL	:
BE	04	66	28	0004A	MOVC3	(R6), @4(R6), @ADDRESS	: 3897
		67	D0	00050	MOVL	CTL\$GL_CLINTOWN, R0	: 3898
		04	A6	9F	PUSHAB	4(R6)	: 3899
		08	AE	9F	PUSHAB	OLD_SIZE	: 3898
	0080	02	FB	00059	CALLS	#2, @128(R0)	: 3899
	04	6E	D0	0005E	MOVL	ADDRESS, 4(R6)	: 3900
		01	D0	00062	MOVL	#1, R0	: 3901
			04	00065	RET		: 3902

; Routine Size: 102 bytes, Routine Base: DCL\$ZCODE + 0FCE

```

2327 3903 1 ROUTINE local_qualifier (entity, number) =
2328 3904 1
2329 3905 1 ---
2330 3906 1
2331 3907 1 Locate the last local occurrence of a qualifier on the command
2332 3908 1 line and return the token descriptor.
2333 3909 1
2334 3910 1 Inputs:
2335 3911 1
2336 3912 1 entity = Address of entity descriptor block
2337 3913 1 number = Qualifier number to search for
2338 3914 1
2339 3915 1 Outputs:
2340 3916 1
2341 3917 1 routine value = Address of token descriptor if found, else 0
2342 3918 1 ---
2343 3919 1
2344 3920 2 BEGIN
2345 3921 2
2346 3922 2 MAP
2347 3923 2 entity: REF BBLOCK;
2348 3924 2
2349 3925 2 BIND
2350 3926 2 wrk = cti$gl_dclprsnw : REF BBLOCK;
2351 3927 2 prmlim = cti$gl_clintown [dcl_l_prm[im] : VECTOR;
2352 3928 2 last_param = cti$gl_clintown [dcl_b_param] : BYTE;
2353 3929 2
2354 3930 2 LOCAL
2355 3931 2 token: REF BBLOCK, ! Address of current token
2356 3932 2 index, ! Index of current token
2357 3933 2 match; ! Address of last matching token
2358 3934 2
2359 3935 2
2360 3936 2 ! If the qualifier has local or positional placement, then search for
2361 3937 2 a parmqual starting at the last parameter value returned until the
2362 3938 2 next parameter value.
2363 3939 2
2364 3940 2 match = false;
2365 3941 2 IF .entity [ent_v_parm]
2366 3942 2 AND .last_param NEQ 0
2367 3943 2 THEN BEGIN
2368 3944 2 LOCAL plm: REF BBLOCK;
2369 3945 2 plm = prmlim [.last_param-1];
2370 3946 2 index = .plm [plm_b_quadesc];
2371 3947 2
2372 3948 2 WHILE .index NEQ 0
2373 3949 2 AND .index LEQU .plm [plm_b_lstdesc]
2374 3950 2 DO BEGIN
2375 3951 2 token = token_desc(.index);
2376 3952 2
2377 3953 2 IF (.token [ptr_v_type] EQL ptr_k_parametr)
2378 3954 2 AND (.token [ptr_b_level] EQL 1)
2379 3955 2 THEN EXITLOOP;
2380 3956 2
2381 3957 2 IF (.token [ptr_v_type] EQL ptr_k_parmqual)
2382 3958 2 AND (.token [ptr_b_number] EQL .number)
2383 3959 2 THEN match = .token;

```

```

! If no match is found, then return false
! If local or position placement,
! and we have recently requested a parm
! Address of parameter limit descriptor
! Limits of last parameter requested
! Start following last value returned
! While range not yet exhausted,
! Get token descriptor
! If a parameter value,
! then stop the parameter search
! If parameter qualifier,
! and its ours,
! Return descriptor of parmqual

```

```

: 2384      3960 4
: 2385      3961 4      index = .index + 1;
: 2386      3962 4      END;
: 2387      3963 4
: 2388      3964 4      END;
: 2389      3965 4
: 2390      3966 4      RETURN .match;
: 2391      3967 4
: 2392      3968 1 END;

```

! Skip to next descriptor

! Return token address or false

003C 00000 LOCAL_QUALIFIER:

				51	00000000G	00	D0	00002	.WORD	Save R2,R3,R4,R5	3903
						55	D4	00009	MOVL	CTL\$GL_CLINTOWN, R1	3927
				50	04	AC	D0	0000B	CLRL	MATCH	3940
	53	05		A0		01	E1	0000F	MOVL	ENTITY, R0	3941
					008F	C1	95	00014	BBC	#1, 5(R0), 4\$	
						4D	13	00018	TSTB	143(R1)	3942
				50	008F	C1	9A	0001A	BEQL	4\$	
				54	FC	A140	DE	0001F	MOVZBL	143(R1), R0	3945
				50	03	A4	9A	00024	MOVAL	-4(R1)[R0], PLM	
						3D	13	00028	MOVZBL	3(PLM), INDEX	3946
	50	02	A4	08		00	ED	0002A	BEQL	4\$	3948
						35	1F	00030	CMPZV	#0, #8, 2(PLM), INDEX	3949
				51	00000000G	00	D0	00032	BLSSU	4\$	
				50		0C	C5	00039	MOVL	WRK, R1	3951
		52		53	F9AA	C241	9E	0003D	MULL3	#12, INDEX, R2	
	03		63	04		1C	ED	00043	MOVAB	-1622(R2)[R1], TOKEN	
						06	12	00048	CMPZV	#28, #4, (TOKEN), #3	3953
				01	04	A3	91	0004A	BNEQ	2\$	
						17	13	0004E	CMPB	4(TOKEN), #1	3954
	01		63	04		1C	ED	00050	BEQL	4\$	
						0C	12	00055	CMPZV	#28, #4, (TOKEN), #1	3957
08	AC	05	A3	08		00	ED	00057	BNEQ	3\$	
						03	12	0005E	CMPZV	#0, #8, 5(TOKEN), NUMBER	3958
				55		53	D0	00060	BNEQ	3\$	
						50	D6	00063	MOVL	TOKEN, MATCH	3959
						C1	11	00065	INCL	INDEX	3961
				50		55	D0	00067	BRB	1\$	3948
						04	0006A		MOVL	MATCH, R0	3966
									RET		3968

; Routine Size: 107 bytes, Routine Base: DCL\$ZCODE + 1034

```

2394 3969 1 ROUTINE global_qualifier (entity, number) =
2395 3970 1
2396 3971 1 ---
2397 3972 1
2398 3973 1 Locate the last global occurrence of a qualifier on the command
2399 3974 1 line and return the token descriptor.
2400 3975 1
2401 3976 1 Inputs:
2402 3977 1
2403 3978 1 entity = Address of entity descriptor block
2404 3979 1 number = Qualifier number to search for
2405 3980 1
2406 3981 1 Outputs:
2407 3982 1
2408 3983 1 routine value = Address of token descriptor if found, else 0
2409 3984 1 ---
2410 3985 1
2411 3986 2 BEGIN
2412 3987 2
2413 3988 2 BIND
2414 3989 2 entity_context = cti$gl_clintown [dcl_l_entity] : VECTOR, ! Entity context array
2415 3990 2 last_qual = cti$gl_clintown [dcl_l_qual], ! Last qualifier token
2416 3991 2 wrk = cti$gl_dclpr$own : REF BBLOCK;
2417 3992 2
2418 3993 2 LOCAL
2419 3994 2 last: REF BBLOCK, ! Address of token for last occurrence
2420 3995 2 token: REF BBLOCK, ! Address of current token
2421 3996 2 index; ! Index of current token
2422 3997 2
2423 3998 2
2424 3999 2 ! If in midst of CLISNEXT_QUAL call, then return the already found global
2425 4000 2 qualifier token.
2426 4001 2
2427 4002 2 IF .cti$gl_clintown [dcl_v_nextqual] AND (.entity_context [0] EQL .entity)
2428 4003 2 THEN RETURN .last_qual;
2429 4004 2
2430 4005 2
2431 4006 2 ! Search for the last occurrence as a command qualifier.
2432 4007 2
2433 4008 2 last = 0; ! Indicate no occurrences found
2434 4009 2 index = 1; ! Start at first token descriptor
2435 4010 2 token = token_desc(1);
2436 4011 2
2437 4012 2 WHILE (.token [ptr_v_type] NEQ ptr_k_endline) ! Until end of command line
2438 4013 2 DO BEGIN
2439 4014 2
2440 4015 2 IF .token [ptr_v_type] EQL ptr_k_comdqual ! If token is a qualifier
2441 4016 2 AND .token [ptr_b_number] EQL .number ! and its our qualifier
2442 4017 2 THEN last = .token; ! Save last occurrence of qualifier
2443 4018 2
2444 4019 2 token = .token + ptr_c_length; ! Skip to next token
2445 4020 2 index = .index + 1; ! and increment token index
2446 4021 2 END;
2447 4022 2
2448 4023 2 RETURN .last; ! Return address of token descriptor
2449 4024 2
2450 4025 1 END;

```

0004 00000 GLOBAL_QUALIFIER:

			50	00000000G	00	D0	00002	.WORD	Save R2	:	3969
	0C	008C	C0		01	E1	00009	MOVL	CTL\$GL CLINTOWN, R0	:	3989
		04	AC	40	A0	D1	0000F	BBC	#1, 140(R0), 1\$	:	4002
					05	12	00014	CMPL	64(R0), ENTITY	:	
			50	78	A0	D0	00016	BNEQ	1\$	:	
						04	0001A	MOVL	120(R0), R0	:	4003
					51	D4	0001B	RET		:	
			52		01	D0	0001D	CLRL	LAST	:	4008
		50	00000000G	00	8F	C3	00020	MOVL	#1, INDEX	:	4009
	04	60		04	1C	ED	0002C	SUBL3	#1610, WRK, TOKEN	:	4010
					1A	13	00031	CMPZV	#28, #4, (TOKEN), #4	:	4012
	00	60		04	1C	ED	00033	BEQL	4\$	:	
					0C	12	00038	CMPZV	#28, #4, (TOKEN), #0	:	4015
08	AC	05	A0	08	00	ED	0003A	BNEQ	3\$	:	
					03	12	00041	CMPZV	#0, #8, 5(TOKEN), NUMBER	:	4016
			51		50	D0	00043	BNEQ	3\$	:	
			50		0C	C0	00046	MOVL	TOKEN, LAST	:	4017
					52	D6	00049	ADDL2	#12, TOKEN	:	4019
					DF	11	0004B	INCL	INDEX	:	4020
			50		51	D0	0004D	BRB	2\$	:	4012
						04	00050	MOVL	LAST, R0	:	4023
								RET		:	4025

; Routine Size: 81 bytes, Routine Base: DCL\$ZCODE + 109F


```

: 2452      4026 1 ROUTINE token_string (token, retdesc): NOVALUE =
: 2453      4027 1
: 2454      4028 1 ---
: 2455      4029 1
: 2456      4030 1 Create a string descriptor of a token string
: 2457      4031 1
: 2458      4032 1 Inputs:
: 2459      4033 1
: 2460      4034 1 token = Address of token descriptor
: 2461      4035 1 retdesc = Address of quadword to receive string descriptor
: 2462      4036 1
: 2463      4037 1 Outputs:
: 2464      4038 1
: 2465      4039 1 retdesc = Descriptor of token string
: 2466      4040 1 ---
: 2467      4041 1
: 2468      4042 2 BEGIN
: 2469      4043 2
: 2470      4044 2 BIND
: 2471      4045 2 wrk = ctl$gl_dclprstown : REF BBLOCK;
: 2472      4046 2
: 2473      4047 2 MAP
: 2474      4048 2 token: REF BBLOCK,
: 2475      4049 2 retdesc: REF VECTOR [2];
: 2476      4050 2
: 2477      4051 2 retdesc [0] = .token [ptr_b_value]; ! Return the token length
: 2478      4052 2 retdesc [1] = wrk [wrk_g_buffer] + .token [ptr_v_offset]; ! Return the token address
: 2479      4053 2
: 2480      4054 1 END;

```

				0000 0000	TOKEN_STRING:			
					.WORD	Save nothing		: 4026
		50	08	AC	D0 00002	MOVL	RETDESC, R0	: 4051
		60	04	BC	9A 00006	MOVZBL	@TOKEN, (R0)	
51	04	0C		08	EF 0000A	EXTZV	#8, #12, @TOKEN, R1	: 4052
		51	00000000G	00	C0 00010	ADDL2	WRK, R1	
	04	A0	F492	C1	9F 00017	MOVAB	-2926(R1), 4(R0)	
				04	0001D	RET		: 4054

; Routine Size: 30 bytes, Routine Base: DCL\$ZCODE + 10F0

```

: 2482 4055 1 ROUTINE upcase (input, output): NOVALUE =
: 2483 4056 1
: 2484 4057 1 ---
: 2485 4058 1
: 2486 4059 1     Upcase a string.
: 2487 4060 1
: 2488 4061 1     Inputs:
: 2489 4062 1
: 2490 4063 1         input = address of input string descriptor
: 2491 4064 1         output = address of output string descriptor
: 2492 4065 1
: 2493 4066 1     Outputs:
: 2494 4067 1
: 2495 4068 1         The string is upcased.
: 2496 4069 1 ---
: 2497 4070 1
: 2498 4071 2 BEGIN
: 2499 4072 2
: 2500 4073 2 MAP
: 2501 4074 2     input : REF BBLOCK,
: 2502 4075 2     output : REF BBLOCK;
: 2503 4076 2
: 2504 4077 2 REGISTER
: 2505 4078 2
: 2506 4079 2     ptr: REF VECTOR [,BYTE],
: 2507 4080 2     char: BYTE;
: 2508 4081 2
: 2509 4082 2 BIND
: 2510 4083 2
: 2511 4084 2     uc_tbl = CH$TRANSTABLE
: 2512 4085 2 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
: 2513 4086 2 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
: 2514 4087 2 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
: 2515 4088 2 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
: 2516 4089 2 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
: 2517 4090 2 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,
: 2518 4091 2 96, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, ! a-o -> A-O.
: 2519 4092 2 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 123, 124, 125, 126, 127, ! p-z -> P-Z.
: 2520 4093 2 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143,
: 2521 4094 2 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159,
: 2522 4095 2 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175,
: 2523 4096 2 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191,
: 2524 4097 2 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
: 2525 4098 2 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223,
: 2526 4099 2 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, ! Upcase foreign
: 2527 4100 2 240, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 254, 255); ! too.
: 2528 4101 2
: 2529 4102 2
: 2530 4103 2     output [dsc$w_length] = .input [dsc$w_length]; ! Use the original string length
: 2531 4104 2     CH$TRANSLATE (uc_tbl, .input [dsc$w_length], .input [dsc$a_pointer], ! Translate characters
: 2532 4105 2         0, -32, .output [dsc$a_pointer]);
: 2533 4106 2
: 2534 4107 1 END;

```


RPCLINT
V04-000

B 3
16-Sep-1984 00:26:36
14-Sep-1984 12:15:33

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCL.SRC]RPCLINT.B32;1 Page 87
(32)

0E	0D	0C	0B	0A	09	08	07	06	05	04	03	02	01	00	011110
1D	1C	1B	1A	19	18	17	16	15	14	13	12	11	10	0F	01111F
2C	2B	2A	29	28	27	26	25	24	23	22	21	20	1F	1E	01112E
3B	3A	39	38	37	36	35	34	33	32	31	30	2F	2E	2D	01113D
4A	49	48	47	46	45	44	43	42	41	40	3F	3E	3D	3C	01114C
59	58	57	56	55	54	53	52	51	50	4F	4E	4D	4C	4B	01115B
48	47	46	45	44	43	42	41	60	5F	5E	5D	5C	5B	5A	01116A
57	56	55	54	53	52	51	50	4F	4E	4D	4C	4B	4A	49	011179
86	85	84	83	82	81	80	7F	7E	7D	7C	7B	7A	79	78	011188
95	94	93	92	91	90	8F	8E	8D	8C	8B	8A	89	88	87	011197
A4	A3	A2	A1	A0	9F	9E	9D	9C	9B	9A	99	98	97	96	0111A6
B3	B2	B1	B0	AF	AE	AD	AC	AB	AA	A9	A8	A7	A6	A5	0111B5
C2	C1	C0	BF	BE	BD	BC	BB	BA	B9	B8	B7	B6	B5	B4	0111C4
D1	D0	CF	CE	CD	CC	CB	CA	C9	C8	C7	C6	C5	C4	C3	0111D3
CO	DF	DE	DD	DC	DB	DA	D9	D8	D7	D6	D5	D4	D3	D2	0111E2
CF	CE	CD	CC	CB	CA	C9	C8	C7	C6	C5	C4	C3	C2	C1	0111F1
FE	DD	DC	DB	DA	D9	D8	D7	D6	D5	D4	D3	D2	D1	F0	01200F
														FF	0120FF

P.AAB: .BYTE

0	1	2	3	4	5	6	7	8	9	10	11	12	-	:							
13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	-	:
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	-	:
53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	-	:
73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	-	:
93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	-	:
113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	-	:
133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	-	:
153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	-	:
173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	-	:
193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	-	:
213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	-	:
233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	-	:
253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	-	:
273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	-	:
293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	-	:
313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	-	:
333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	-	:
353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	-	:
373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	-	:
393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	-	:
413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	-	:
433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	-	:
453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	-	:
473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	-	:
493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	-	:
513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	-	:
533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	-	:
553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	-	:
573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	-	:
593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	-	:
613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	-	:
633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	-	:
653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	-	:
673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	-	:
693	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	-	:
713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	-	:
733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	-	:
753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	-	:
773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	-	:
793	794	795	796	797	798	799	800	801	802	803	804	805	806	807	808	809	810	811	812	-	:
813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	-	:
833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	-	:
853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	-	:
873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	-	:
893	894	895	896	897	898	899	900	901	902	903	904	905	906	907	908	909	910	911	912	-	:
913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	-	:
933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	-	:
953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	-	:
973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	-	:
993	994	995	996	997	998	999	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009	1010	1011	1012	-	:
1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023	1024	1025	1026	1027	1028	1029	1030	1031	1032	-	:
1033	1034	1035	1036	1037	1038	1039	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	-	:
1053	1054	1055	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071	1072	-	:
1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087	1088	1089	1090	1091	1092	-	:
1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103	1104	1105	1106	1107	1108	1109	1110	1111	1112	-	:
1113	1114	1115	1116	1117	1118	1119	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	-	:
1133	1134	1135	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151	1152	-	:
1153	1154	1155	1156	1157	1158																

```

: 2536 4108 1 ROUTINE convert_keyword_list (desc, array) =
: 2537 4109 1
: 2538 4110 1 ---
: 2539 4111 1
: 2540 4112 1 Take the user's string apart and fill in the keyword array.
: 2541 4113 1
: 2542 4114 1 Inputs:
: 2543 4115 1
: 2544 4116 1 desc = Address of descriptor of user's input string
: 2545 4117 1 array = Address of the array of descriptors to be filled in
: 2546 4118 1
: 2547 4119 1 Outputs:
: 2548 4120 1
: 2549 4121 1 The array is set up.
: 2550 4122 1 An error code is returned if there is a syntax error in the input string.
: 2551 4123 1 The error is signalled here.
: 2552 4124 1 ---
: 2553 4125 1
: 2554 4126 2 BEGIN
: 2555 4127 2
: 2556 4128 2 MAP
: 2557 4129 2 desc : REF BBLOCK,
: 2558 4130 2 array : REF VECTOR;
: 2559 4131 2
: 2560 4132 2 LOCAL
: 2561 4133 2 ptr,
: 2562 4134 2 old_ptr,
: 2563 4135 2 index,
: 2564 4136 2 status;
: 2565 4137 2
: 2566 4138 2 CH$FILL (0, 4*(2*(dcl_c_context+1)+1), .array);
: 2567 4139 2 ptr = old_ptr = .desc [dsc$a_pointer];
: 2568 4140 2 index = 0;
: 2569 4141 2 status = false;
: 2570 4142 2
: 2571 4143 4 WHILE ((.ptr LSSU .desc [dsc$a_pointer] + .desc [dsc$w_length])
: 2572 4144 3 AND (.index LSSU 2*(dcl_c_context+1)))
: 2573 4145 3 DO BEGIN
: 2574 4146 3 ptr = CH$FIND_CH (.desc [dsc$a_pointer] + .desc [dsc$w_length] - .ptr,
: 2575 4147 3 .ptr, %C');
: 2576 4148 3 IF .ptr EQL 0
: 2577 4149 3 THEN EXITLOOP status = true;
: 2578 4150 3 array [.index] = .ptr - .old_ptr;
: 2579 4151 3 array [.index + 1] = .old_ptr;
: 2580 4152 3 ptr = .ptr + 1;
: 2581 4153 3 old_ptr = .ptr;
: 2582 4154 3 index = .index + 2;
: 2583 4155 2 END;
: 2584 4156 2
: 2585 4157 2 IF NOT .status
: 2586 4158 2 THEN BEGIN
: 2587 4159 2 SIGNAL (msg$_noentity, 1, .desc, cli$_entnf);
: 2588 4160 2 RETURN msg$_noentity
: 2589 4161 2 END;
: 2590 4162 2
: 2591 4163 2 array [.index] = .desc [dsc$a_pointer] + .desc [dsc$w_length] - .old_ptr;
: 2592 4164 2 array [.index + 1] = .old_ptr;

```

: 2593
: 2594
4165 2 RETURN true;
4166 1 END;

007C 00000 CONVERT_KEYWORD_LIST:

0044	8F	00	56	08	AC	D0	00002	WORD	Save R2,R3,R4,R5,R6	4108
			6E		00	2C	00006	MOVL	ARRAY, R6	4138
					66		0000D	MOVCS	#0, (SP), #0, #68, (R6)	
			53	04	AC	D0	0000E	MOVL	DESC, R3	4139
			54	04	A3	D0	00012	MOVL	4(R3), OLD_PTR	
			51		54	D0	00016	MOVL	OLD_PTR, PTR	
					52	D4	00019	CLRL	INDEX	4140
					55	D4	0001B	CLRL	STATUS	4141
			50		63	3C	0001D	MOVZWL	(R3), R0	4143
			50	04	A3	C0	00020	ADDL2	4(R3), R0	
			50		51	D1	00024	CMPL	PTR, R0	
					2D	1E	00027	BGEQU	4\$	
			10		52	D1	00029	CMPL	INDEX, #16	4144
					28	1E	0002C	BGEQU	4\$	
			50		51	C2	0002E	SUBL2	PTR, R0	4146
		61	50		2E	3A	00031	LOCC	#46, R0, (PTR)	
					02	12	00035	BNEQ	2\$	
					51	D4	00037	CLRL	R1	
					51	D5	00039	TSTL	PTR	4148
					05	12	0003B	BNEQ	3\$	
			55		01	D0	0003D	MOVL	#1, STATUS	4149
					14	11	00040	BRB	4\$	
		6642	51		54	C3	00042	SUBL3	OLD_PTR, PTR, (R6)[INDEX]	4150
			04 A642		54	D0	00047	MOVL	OLD_PTR, 4(R6)[INDEX]	4151
					51	D6	0004C	INCL	PTR	4152
			54		51	D0	0004E	MOVL	PTR, OLD_PTR	4153
			52		02	C0	00051	ADDL2	#2, INDEX	4154
					C7	11	00054	BRB	1\$	4143
			1F		55	E8	00056	BLBS	STATUS, 5\$	4157
					8F	DD	00059	PUSHL	#CLIS_ENTNF	4159
					53	DD	0005F	PUSHL	R3	
					01	DD	00061	PUSHL	#1	
					8F	DD	00063	PUSHL	#200956	
			00000000G		04	FB	00069	CALLS	#4, LIB\$SIGNAL	
					50	000310FC	8F	D0	00070	4160
					04	00077	04	00077	RET	
			50		63	3C	00078	MOVZWL	(R3), R0	4163
			50	04	A3	C1	0007B	ADDL3	4(R3), R0, R3	
			53		54	C3	00080	SUBL3	OLD_PTR, R3, (R6)[INDEX]	
			04 A642		54	D0	00085	MOVL	OLD_PTR, 4(R6)[INDEX]	4164
			50		01	D0	0008A	MOVL	#1, R0	4165
					04	0008D	04	0008D	RET	4166

; Routine Size: 142 bytes, Routine Base: DCL\$ZCODE + 1225

```

: 2596 4167 1 ROUTINE batch_job =
: 2597 4168 1
: 2598 4169 1 ---
: 2599 4170 1
: 2600 4171 1 This routine returns a boolean value indicating whether
: 2601 4172 1 the current process is a batch job or not.
: 2602 4173 1
: 2603 4174 1 Inputs:
: 2604 4175 1
: 2605 4176 1 None
: 2606 4177 1
: 2607 4178 1 Outputs.
: 2608 4179 1
: 2609 4180 1 Routine value is true if a batch job, else false
: 2610 4181 1 ---
: 2611 4182 1
: 2612 4183 2 BEGIN
: 2613 4184 2
: 2614 4185 2 LOCAL
: 2615 4186 2 pcb_sts: BBLOCK [4],
: 2616 4187 2 item_list: BBLOCK [16],
: 2617 4188 2 iosb: BBLOCK [8],
: 2618 4189 2 status;
: 2619 4190 2
: 2620 4191 2
: 2621 4192 2 Get job status flags to determine type of job
: 2622 4193 2
: 2623 4194 2 item_list [0,0,16,0] = 4;
: 2624 4195 2 item_list [2,0,16,0] = jpi$sts;
: 2625 4196 2 item_list [4,0,32,0] = pcb_sts;
: 2626 4197 2 item_list [8,0,32,0] = 0;
: 2627 4198 2 item_list [12,0,32,0] = 0;
: 2628 4199 2
: 2629 4200 2 iosb [0,0,32,0] = 0;
: 2630 4201 2 iosb [4,0,32,0] = 0;
: 2631 4202 2
: 2632 P 4203 2 return_if_error ($GETJPIW (ITMLST = item_list,
: 2633 P 4204 2 EFN = exe$csysefn,
: 2634 4205 2 IOSB = iosb));
: 2635 4206 2
: 2636 4207 3 IF NOT (status = .iosb [0,0,16,0])
: 2637 4208 2 THEN RETURN .status;
: 2638 4209 2
: 2639 4210 2 RETURN .pcb_sts <$BITPOSITION(pcb$batch),1>;
: 2640 4211 2
: 2641 4212 1 END;

```

```

! PCB status flags
! GETJPI item list
! IOSB
! Status

! Buffer length
! JPI code
! Buffer address
! Address to return item length
! End of item list

! Init the IOSB

! Obtain PCB flags

! Return errant IOSB status codes

! True if batch job

```

.EXTRN SYS\$GETJPIW

0000 00000 BATCH_JOB:

```

OC SE 1C C2 00002
10 AE 03050004 8F D0 00005
6E 9E 0000D
14 AE 7C 00011

```

```

.WORD Save nothing
SUBL2 #28, SP
MOVL #50659332, ITEM_LIST
MOVAB PCB_STS, ITEM_LIST+4
CLRQ ITEM_LIST+8

```

```

: 4167
: 4194
: 4196
: 4197

```

RPCLINT
V04-000

F 3
16-Sep-1984 00:26:36
14-Sep-1984 12:15:33

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCL.SRC]RPCLINT.B32;1
Page 91
(34)

			04	AE	7C	00014	CLRQ	IOSB	:	4200
				7E	7C	00017	CLRQ	-(SP)	:	4205
			0C	AE	9F	00019	PUSHAB	IOSB	:	
			18	AE	9F	0001C	PUSHAB	ITEM_LIST	:	
				7E	7C	0001F	CLRQ	-(SPT)	:	
		00000000G	00	8F	DD	00021	PUSHL	#EXESC SYSEFN	:	
			0D	07	FB	00027	CALLS	#7, SYSSGETJPIW	:	
			50	50	E9	0002E	BLBC	STATUS, 1\$	:	
			06	04	AE	3C	MOVZWL	IOSB, STATUS	:	4207
50	01	AE	06	50	E9	00035	BLBC	STATUS, 1\$	:	
			01	06	EF	00038	EXTZV	#6, #1, PCB_STS+1, R0	:	4210
				04	0003E	1\$:	RET		:	4212

; Routine Size: 63 bytes, Routine Base: DCL\$ZCODE + 12B3

```

2643 4213 1 GLOBAL ROUTINE dcl$dispatch (rqdesc, rqwork, rqbits) =
2644 4214 1
2645 4215 1 ---
2646 4216 1
2647 4217 1 This routine can be called to dispatch to any verb processing
2648 4218 1 routines if the command has the ROUTINE attribute.
2649 4219 1
2650 4220 1 Inputs:
2651 4221 1
2652 4222 1 rqdesc = Address of request descriptor data structure
2653 4223 1 rqword, rqbits = ignored
2654 4224 1
2655 4225 1 Outputs:
2656 4226 1
2657 4227 1 The verb routine is called (if any).
2658 4228 1
2659 4229 1 The status passed back from the routine is returned in R0.
2660 4230 1 If no routine is specified, success is returned.
2661 4231 1 ---
2662 4232 1
2663 4233 2 BEGIN
2664 4234 2
2665 4235 2 MAP
2666 4236 2 rqdesc : REF BBLOCK;
2667 4237 2
2668 4238 2 BUILTIN
2669 4239 2 PROBER; ! True if location can be read
2670 4240 2
2671 4241 2 BIND
2672 4242 2 wrk = ctl$gl_dclprstown : REF BBLOCK; ! Address of command work area
2673 4243 2
2674 4244 2 LOCAL
2675 4245 2 ptr; ! Pointer to offset to user routine
2676 4246 2
2677 4247 2 IF .ctl$gl_clintown EQL 0 ! If not yet initialized,
2678 4248 2 THEN initialize (.rqdesc [int_l_getvm], ! then initialize parsing
2679 4249 2 .rqdesc [int_l_freevm]);
2680 4250 2
2681 4251 2 IF .wrk [wrk_v_userrtn] AND (.wrk [wrk_l_image] NEQ 0) ! If addr of user routine
2682 4252 2 THEN BEGIN ! Then call it
2683 4253 2 ptr = .wrk [wrk_l_image]; ! Get pointer to offset longword
2684 4254 2 IF PROBER(%REF(psT$sc_user), %REF(ptr_c_length), .ptr) ! If location can be read,
2685 4255 2 THEN IF .rqdesc [int_l_entaddr] NEQ 0 ! If user-supplied argument
2686 4256 2 THEN RETURN (.ptr)(.rqdesc [int_l_entaddr]) ! then call user routine with argument
2687 4257 2 ELSE RETURN (.ptr)(); ! else call user routine without argument
2688 4258 2 END;
2689 4259 2
2690 4260 2 SIGNAL (cli$ invrout); ! Signal error
2691 4261 2 RETURN cli$_invrout; ! Return error
2692 4262 1 END;

```

52 00000000G 8F 00 00002

.ENTRY DCL\$DISPATCH, Save R2
MOVL #CLIS_INVROUT, R2

: 4213
:

		00000000G	00	D5	00009	TSTL	CTL\$GL_CLINTOWN	:	4247
			0D	12	0000F	BNEQ	1\$	:	
	50	04	AC	D0	00011	MOVL	RQDESC, R0	:	4249
	7E	10	A0	7D	00015	MOVQ	16(R0), -(SP)	:	4248
	ECFD		02	FB	00019	CALLS	#2, INITIALIZE	:	
23		00000000G	00	D0	0001E	1\$: MOVL	WRK, R0	:	4251
	F2		01	E1	00025	BBC	#1, -14(R0), 3\$	:	
			E2	A0	D5 0002A	TSTL	-30(R0)	:	
			1E	13	0002D	BEQL	3\$	:	
	51		E2	A0	D0 0002F	MOVL	-30(R0), PTR	:	4253
61			03	0C	00033	PROBER	#3, #12, (PTR)	:	4254
			14	13	00037	BEQL	3\$	:	
	50	04	AC	D0	00039	MOVL	RQDESC, R0	:	4255
		0C	A0	D5	0003D	TSTL	12(R0)	:	
			07	13	00040	BEQL	2\$	:	
		0C	A0	DD	00042	PUSHL	12(R0)	:	4256
	61		01	FB	00045	CALLS	#1, (PTR)	:	
				04	00048	RET		:	4257
	61		00	FB	00049	2\$: CALLS	#0, (PTR)	:	
				04	0004C	RET		:	
			52	DD	0004D	3\$: PUSHL	R2	:	4260
	00000000G	00	01	FB	0004F	CALLS	#1, LIB\$SIGNAL	:	
		50	52	D0	00056	MOVL	R2, R0	:	4261
				04	00059	RET		:	4262

; Routine Size: 90 bytes, Routine Base: DCL\$ZCODE + 12F2

```

: 2694 4263 1 GLOBAL ROUTINE dcl$nextqual (rqdesc, rqwork, rqbits) =
: 2695 4264 1
: 2696 4265 1 ---
: 2697 4266 1
: 2698 4267 1 Point to the next instance of the specified qualifier
: 2699 4268 1 in the command line.
: 2700 4269 1
: 2701 4270 1 Inputs:
: 2702 4271 1
: 2703 4272 1 rqdesc = Address of request descriptor data structure
: 2704 4273 1 rqword, rqbits = ignored
: 2705 4274 1
: 2706 4275 1 Outputs:
: 2707 4276 1
: 2708 4277 1 Routine value:
: 2709 4278 1
: 2710 4279 1 success = cli$_present
: 2711 4280 1 cli$_locpres
: 2712 4281 1 cli$_defaulted
: 2713 4282 1
: 2714 4283 1 failure = cli$_absent
: 2715 4284 1 cli$_negated
: 2716 4285 1 cli$_locneg
: 2717 4286 1
: 2718 4287 1 All errors are signalled.
: 2719 4288 1 ---
: 2720 4289 1
: 2721 4290 2 BEGIN
: 2722 4291 2
: 2723 4292 2 MAP
: 2724 4293 2 rqdesc : REF BBLOCK;
: 2725 4294 2
: 2726 4295 2 BIND
: 2727 4296 2 entity_context = cti$gl_clintown [dcl_l_entity] : VECTOR,
: 2728 4297 2 token_context = cti$gl_clintown [dcl_l_token] : VECTOR,
: 2729 4298 2 last_qual = cti$gl_clintown [dcl_l_qual],
: 2730 4299 2 wrk = cti$gl_dclpr$own : REF BBLOCK;
: 2731 4300 2
: 2732 4301 2 GLOBAL REGISTER
: 2733 4302 2 block=9: REF BBLOCK,
: 2734 4303 2 number=10,
: 2735 4304 2 type=11;
: 2736 4305 2
: 2737 4306 2 LOCAL
: 2738 4307 2 token : REF BBLOCK,
: 2739 4308 2 keyword_array : VECTOR [2*(dcl_c_context+1)+1];
: 2740 4309 2
: 2741 4310 2
: 2742 4311 2 Initialize CLINT if necessary.
: 2743 4312 2
: 2744 4313 2 IF .cti$gl_clintown EQL 0
: 2745 4314 2 THEN initialize (.rqdesc [int_l_getvm],
: 2746 4315 2 .rqdesc [int_l_freevm]);
: 2747 4316 2
: 2748 4317 2
: 2749 4318 2 Verify that valid entities were specified.
: 2750 4319 2

```

```

! Entity context array
! Token context array
! Last qualifier token
! Address of command work area
! Address of entity descriptor block
! Parameter/qualifier number
! Entity type
! Ptr to token descriptor
! Keyword array
! If not yet initialized,
! then initialize parsing

```



```

: 2751 P 4320 2 return_if_error (verify_entities (rqdesc [int_w_entlen],
: 2752 4321 2 keyword_array));
: 2753 4322 2
: 2754 4323 2
: 2755 4324 2 If the entity is not a qualifier then return an error.
: 2756 4325 2
: 2757 4326 2 IF (.type NEQ qual_entity) OR
: 2758 4327 2 (.keyword_array [2] NEQ 0)
: 2759 4328 2 THEN BEGIN
: 2760 4329 2 SIGNAL (msg$_noentity,1,keyword_array [0],cli$_entnf); ! Then signal the error
: 2761 4330 2 RETURN msg$_noentity; ! Return the status
: 2762 4331 2 END;
: 2763 4332 2
: 2764 4333 2
: 2765 4334 2 Do we have a previous context? If so, start search there.
: 2766 4335 2
: 2767 4336 2 IF .entity_context [0] EQL .block
: 2768 4337 2 THEN token = .last_qual + ptr_c_length
: 2769 4338 2 ELSE token = token_desc(1);
: 2770 4339 2 zero_context_arrays [0];
: 2771 4340 2 last_qual = 0;
: 2772 4341 2
: 2773 4342 2
: 2774 4343 2 Search for the next occurrence as a command qualifier.
: 2775 4344 2
: 2776 4345 3 WHILE (.token [ptr_v_type] NEQ ptr_k_endline) ! Until end of command line
: 2777 4346 3 DO BEGIN
: 2778 4347 3
: 2779 4348 4 IF (.token [ptr_v_type] EQL ptr_k_comdqual) ! If token is a qualifier
: 2780 4349 4 AND (.token [ptr_b_number] EQL .number) ! and its our qualifier
: 2781 4350 4 THEN BEGIN ! Save last occurrence of qualifier
: 2782 4351 4 last_qual = .token;
: 2783 4352 4 entity_context [0] = .block;
: 2784 4353 4 ctl$gl_clintown [dcl_v_nextqual] = true; ! Set nextqual qualifier parse
: 2785 4354 4 RETURN cli$_present;
: 2786 4355 4 END;
: 2787 4356 3
: 2788 4357 3 token = .token + ptr_c_length; ! Skip to next token
: 2789 4358 3 END;
: 2790 4359 2
: 2791 4360 2 ctl$gl_clintown [dcl_v_nextqual] = false; ! Set normal qualifier parse
: 2792 4361 2 RETURN cli$_absent; ! Return address of token descriptor
: 2793 4362 1 END;

```

		OFFC 00000	.ENTRY	DCL\$NEXTQUAL, Save R2,R3,R4,R5,R6,R7,R8,R9,-;	4263
				R10,R11	
5E	BC	AE 9E 00002	MOVAB	-68(SP), SP	
50	00000000G	00 D0 00006	MOVL	CTL\$GL_CLINTOWN, R0	4296
58	40	A0 9E 0000D	MOVAB	64(R0), R8	
57	5C	A0 9E 00011	MOVAB	92(R0), R7	4297
56	78	A0 9E 00015	MOVAB	120(R0), R6	4298
		50 D5 00019	TSTL	R0	4313
		0D 12 0001B	BNEQ	1\$	

		50	04	AC	D0	0001D	MOVL	RQDESC, RC	4315
		7E	10	A0	7D	00021	MOVQ	16(R0), -(SP)	4314
	EC97	CF		02	FB	00025	CALLS	#2, INITIALIZE	
				5E	DD	0002A	PUSHL	SP	4321
7E	04	AC		08	C1	0002C	ADDL3	#8, RQDESC, -(SP)	
	F29C	CF		02	FB	00031	CALLS	#2, VERIFY_ENTITIES	
		01		50	E8	00036	BLBS	STATUS, 2\$	
					04	00039	RET		
		02		5B	D1	0003A	CMPL	TYPE, #2	4326
				05	12	0003D	BNEQ	3\$	
			08	AE	D5	0003F	TSTL	KEYWORD_ARRAY+8	4327
				20	13	00042	BEQL	4\$	
			00000000G	8F	DD	00044	PUSHL	#CLIS_ENTNF	4329
			04	AE	9F	0004A	PUSHAB	KEYWORD_ARRAY	
				01	DD	0004D	PUSHL	#1	
			000310FC	8F	DD	0004F	PUSHL	#200956	
	00000000G	00		04	FB	00055	CALLS	#4, LIBSSIGNAL	
		50	000310FC	8F	D0	0005C	MOVL	#200956, R0	4330
					04	00063	RET		
		59		68	D1	00064	CMPL	(R8), BLOCK	4336
				06	12	00067	BNEQ	5\$	
	5B	66		0C	C1	00069	ADDL3	#12, (R6), TOKEN	4337
				0C	11	0006D	BRB	6\$	
	5B	00000000G	00	0000064A	8F	C3	0006F	SUBL3	#1610, WRK, TOKEN
1C	00	6E		00	2C	0007B	MOVCS	#0, (SP), #0, #28, (R8)	4338
				68		00080			4339
1C	00	6E		00	2C	00081	MOVCS	#0, (SP), #0, #28, (R7)	
				67		00086			
				66	D4	00087	CLRL	(R6)	4340
04	6B	04		1C	ED	00089	CMPZV	#28, #4, (TOKEN), #4	4345
				2E	13	0008E	BEQL	9\$	
00	6B	04		1C	ED	00090	CMPZV	#28, #4, (TOKEN), #0	4348
				22	12	00095	BNEQ	8\$	
5A	05	AB	08	00	ED	00097	CMPZV	#0, #8, 5(TOKEN), NUMBER	4349
				1A	12	0009D	BNEQ	8\$	
		66		5B	D0	0009F	MOVL	TOKEN, (R6)	4351
		68		59	D0	000A2	MOVL	BLOCK, (R8)	4352
		50	00000000G	00	D0	000A5	MOVL	CTL\$GL CLINTOWN, R0	4353
	008C	C0		02	88	000AC	BISB2	#2, 140(R0)	
		50	00000000G	8F	D0	000B1	MOVL	#CLIS_PRESENT, R0	4354
					04	000B8	RET		
		5B		0C	C0	000B9	ADDL2	#12, TOKEN	4357
				CB	11	000BC	BRB	7\$	4345
	008C	50	00000000G	00	D0	000BE	MOVL	CTL\$GL CLINTOWN, R0	4360
		C0		02	8A	000C5	BICB2	#2, 140(R0)	
		50	00000000G	8F	D0	000CA	MOVL	#CLIS_ABSENT, R0	4361
					04	000D1	RET		4362

; Routine Size: 210 bytes, Routine Base: DCL\$ZCODE + 134C

```

: 2795 4363 1 GLOBAL ROUTINE dcl$endparse (rqdesc, rqwork, rqbits) =
: 2796 4364 1
: 2797 4365 1 ---
: 2798 4366 1
: 2799 4367 1 This routine is called when the user has completed
: 2800 4368 1 all command line parsing. It checks that all qualifiers
: 2801 4369 1 which appeared on the command line were processed in one
: 2802 4370 1 way or another by the utility.
: 2803 4371 1
: 2804 4372 1 Inputs:
: 2805 4373 1
: 2806 4374 1 rqdesc = Address of request descriptor data structure
: 2807 4375 1 rqword, rqbits = ignored
: 2808 4376 1
: 2809 4377 1 Outputs:
: 2810 4378 1
: 2811 4379 1 None
: 2812 4380 1 ---
: 2813 4381 1
: 2814 4382 2 BEGIN
: 2815 4383 2
: 2816 4384 2 BUILTIN
: 2817 4385 2 PROBEW; ! True if location writable
: 2818 4386 2
: 2819 4387 2 MAP
: 2820 4388 2 rqdesc : REF BBLOCK;
: 2821 4389 2
: 2822 4390 2
: 2823 4391 2 If clint own storage is allocated, then deallocate it.
: 2824 4392 2
: 2825 4393 2 IF .ctl$gl_clintown NEQ 0
: 2826 4394 3 THEN (.rqdesc [int_l_freem])
: 2827 4395 2 (%REF(dcl_c_size), ctl$gl_clintown);
: 2828 4396 2 ctl$gl_clintown = 0;
: 2829 4397 2
: 2830 4398 2
: 2831 4399 2 If user mode WRK area, then deallocate it.
: 2832 4400 2 Zero pointer no matter what mode WRK area is.
: 2833 4401 2
: 2834 4402 2 IF .ctl$gl_dclprstown NEQ 0
: 2835 4403 2 THEN IF PROBEW(%REF(psl$sc_user), %REF(-wrk_k_length), .ctl$gl_dclprstown)
: 2836 4404 3 THEN (.rqdesc [int_l_freem])
: 2837 4405 2 (%REF(-wrk_k_length), ctl$gl_dclprstown);
: 2838 4406 2 ctl$gl_dclprstown = 0;
: 2839 4407 2
: 2840 4408 2 RETURN true;
: 2841 4409 1 END;

```

```

53 00000000G 00 000C 00000
52 00000000G 00 9E 00002
5E 04 C2 00010
63 D5 00013

```

```

.ENTRY DCL$ENDPARSE, Save R2,R3
MOVAB CTL$GL_CLINTOWN, R3
MOVAB CTL$GL_DCLPRSTOWN, R2
SUBL2 #4, SP
TSTL CTL$GL_CLINTOWN

```

```

: 4363
:
:
: 4393

```

RPCLINT
V04-000

M 3
16-Sep-1984 00:26:36
14-Sep-1984 12:15:33

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCL.SRC]RPCLINT.B32;1 Page 98
(37)

				12	13	00015	BEQL	1\$	:	
		50	04	AC	D0	00017	MOVL	RQDESC, R0	:	4394
				53	DD	0001B	PUSHL	R3	:	4395
	04	AE	90	8F	9A	0001D	MOVZBL	#144, 4(SP)	:	
			04	AE	9F	00022	PUSHAB	4(SP)	:	
	1	B0		02	FB	00025	CALLS	#2, @20(R0)	:	
				63	D4	00029	CLRL	CTL\$GL_CLINTOWN	:	4396
		50		62	D0	0002B	MOVL	CTL\$GL_DCLPRSOWN, R0	:	4402
				1B	13	0002E	BEQL	2\$	:	
60	0B7A	8F		03	0D	00030	PROBEW	#3, #2938, (R0)	:	4403
				13	13	00036	BEQL	2\$	:	
		50	04	AC	D0	00038	MOVL	RQDESC, R0	:	4404
				52	DD	0003C	PUSHL	R2	:	4405
	04	AE	0B7A	8F	3C	0003E	MOVZWL	#2938, 4(SP)	:	
			04	AE	9F	00044	PUSHAB	4(SP)	:	
	14	B0		02	FB	00047	CALLS	#2, @20(R0)	:	
				62	D4	0004B	CLRL	CTL\$GL_DCLPRSOWN	:	4406
		50		01	D0	0004D	MOVL	#1, R0	:	4408
				04	00050		RET		:	4409

; Routine Size: 81 bytes, Routine Base: DCL\$ZCODE + 141E

```

2843 4410 1 GLOBAL ROUTINE dcl$getline (rqdesc, rqwork, rqbits) =
2844 4411 1
2845 4412 1 ---
2846 4413 1
2847 4414 1 This routine is called to obtain the complete command line,
2848 4415 1 including the verb.
2849 4416 1
2850 4417 1 Inputs:
2851 4418 1
2852 4419 1 rqdesc = Address of request descriptor data structure
2853 4420 1 rqword, rqbits = ignored
2854 4421 1
2855 4422 1 Outputs:
2856 4423 1
2857 4424 1 The command line is returned via the quadword descriptor
2858 4425 1 contained within the request descriptor block.
2859 4426 1
2860 4427 1 Routine always returns true status.
2861 4428 1 ---
2862 4429 1
2863 4430 2 BEGIN
2864 4431 2
2865 4432 2 MAP
2866 4433 2 rqdesc : REF BBLOCK;
2867 4434 2
2868 4435 2 LOCAL
2869 4436 2 req_desc : BBLOCK [cli$c_reqdesc],
2870 4437 2 rpw : BBLOCK [cli$c_workarea],
2871 4438 2 req_flags : BITVECTOR [32],
2872 4439 2 token : REF BBLOCK,
2873 4440 2 wrk : REF BBLOCK;
2874 4441 2
2875 4442 2 CH$FILL (0,cli$c_reqdesc,req_desc);
2876 4443 2 req_desc [cli$b_rqtype] = cli$k_initprs;
2877 4444 2 SYS$CLI (req_desc, rpw,req_flags);
2878 4445 2
2879 4446 2 wrk = .rpw [rpw_l_dclwrk];
2880 4447 2 token = wrk [wrk_g_result];
2881 4448 2
2882 4449 3 WHILE (.token [ptr_v_type] NEQ ptr_k_endline)
2883 4450 2 DO token = .token + ptr_c_length;
2884 4451 2
2885 4452 2 rqdesc [int_w_entlen] = .token [ptr_v_offset];
2886 4453 2 rqdesc [int_l_entaddr] = wrk [wrk_g_buffer];
2887 4454 2
2888 4455 2 IF CH$RCHAR (.rqdesc [int_l_entaddr]) EQL %C'$'
2889 4456 3 THEN BEGIN
2890 4457 3 rqdesc [int_w_entlen] = .rqdesc [int_w_entlen] - 1;
2891 4458 3 rqdesc [int_l_entaddr] = .rqdesc [int_l_entaddr] + 1;
2892 4459 2 END;
2893 4460 2
2894 4461 2 RETURN true;
2895 4462 1 END;

```

```

! Callback request descriptor
! Result parse work area
! Callback request flags

! Zero request desc block
! Set request type
! Init result parsing solely
! to get rpw [rpw_l_dclwrk]
! Get address of wrk area
! Start at first token descriptor

! Until end of command line
! then skip to next one

! Line length is offset to eol
! and set address of input buffer

! If line is preceeded with '$'
! then strip it off

```

1C	00	5E	FF60	CE	003C	00000	.ENTRY	DCL\$GETLINE, Save R2,R3,R4,R5	:	4410
		6E		9E	00	00002	MOVAB	-160(SP), SP	:	
			E4	AD	2C	00007	MOVCS	#0, (SP), #0, #28, REQ_DESC	:	4442
			E4	AD	94	0000C	CLRB	REQ_DESC	:	4443
			SE	DD	00	00011	PUSHL	SP	:	4444
			08	AE	9F	00013	PUSHAB	RPW	:	
			E4	AD	9F	00016	PUSHAB	REQ_DESC	:	
	00000000G	00		03	FB	00019	CALLS	#3, -SYSSCLI	:	
		51	08	AE	D0	00020	MOVL	RPW+4, WRK	:	4446
04	60	50	F9B6	C1	9E	00024	MOVAB	-1610(R1), TOKEN	:	4447
		04		1C	ED	00029	CMPZV	#28, #4, (TOKEN), #4	:	4449
				05	13	0002E	BEQL	2\$	:	
		50		0C	C0	00030	ADDL2	#12, TOKEN	:	4450
				F4	11	00033	BRB	1\$	:	
		52	04	AC	D0	00035	MOVL	RQDESC, R2	:	4452
53	01	0C		00	EF	00039	EXTZV	#0, #12, 1(TOKEN), R3	:	
		08		53	B0	0003F	MOVW	R3, 8(R2)	:	
		0C		C1	9E	00043	MOVAB	-2926(R1), 12(R2)	:	4453
		24	F492	B2	91	00049	CMPB	@12(R2), #36	:	4455
				06	12	0004D	BNEQ	3\$	:	
			08	A2	B7	0004F	DECW	8(R2)	:	4457
			0C	A2	D6	00052	INCL	12(R2)	:	4458
		50		01	D0	00055	MOVL	#1, R0	:	4461
				04	00	00058	RET		:	4462

; Routine Size: 89 bytes, Routine Base: DCL\$ZCODE + 146F

RPCLINT
V04-000

C 4
16-Sep-1984 00:26:36
14-Sep-1984 12:15:33

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCL.SRC]RPCLINT.B32;1 (39)

Page 101

: 2897 4463 1 END
: 2898 4464 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

:
: Name Bytes Attributes
: DCL\$ZCODE 5320 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(0)

Library Statistics

:
: File ----- Symbols ----- Pages Processing
: Total Loaded Percent Mapped Time
: _\$255\$DUA28:[SYSLIB]LIB.L32;1 18619 24 0 1000 00:01.8

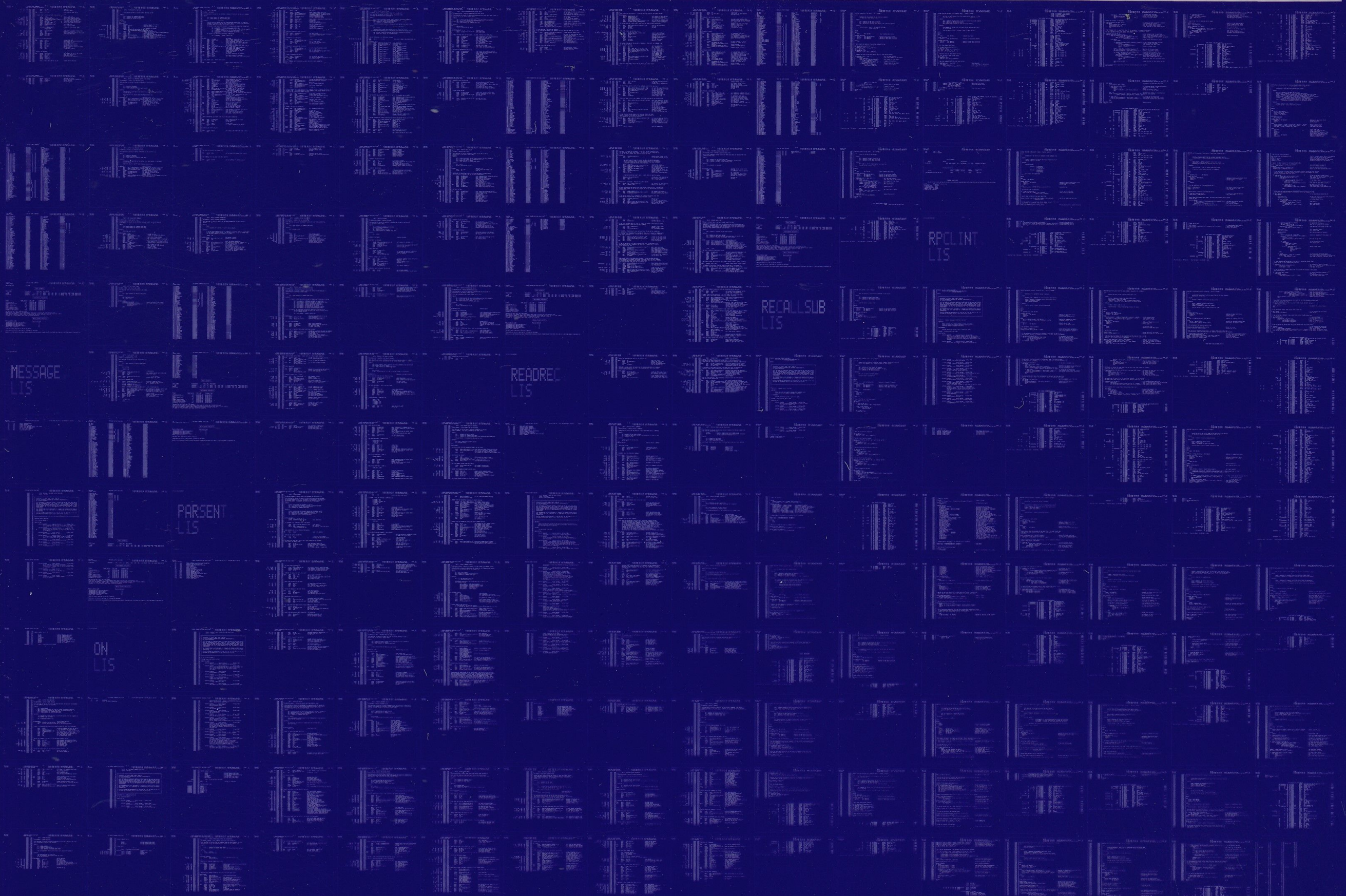
COMMAND QUALIFIERS

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

: Size: 5049 code + 271 data bytes
: Run Time: 01:37.5
: Elapsed Time: 05:04.5
: Lines/CPU Min: 2745
: Lexemes/CPU-Min: 23002
: Memory Used: 356 pages
: Compilation Complete

0072 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0073 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

