

CCCCCCCCCCCC	DDDDDDDDDDDD	UUU	UUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUU	UUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU

```

LL          IIIII
LL          IIIII
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LLLLLLLLLLL IIIII
LLLLLLLLLLL IIIII
SSSSSSSSS
SSSSSSSSS
SS
SS
SS
SS
SSSSSS
SSSSSS
SS
SS
SS
SS
SSSSSSSSS
SSSSSSSSS

```



```
1 0001 0 MODULE gencode3 (IDENT='V04-000'
2 0002 0 ADDRESSING_MODE(EXTERNAL=GENERAL))
3 0003 1 = BEGIN
4 0004 1
5 0005 1 *****
6 0006 1 *
7 0007 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
8 0008 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
9 0009 1 * ALL RIGHTS RESERVED.
10 0010 1 *
11 0011 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
12 0012 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
13 0013 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
14 0014 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
15 0015 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
16 0016 1 * TRANSFERRED.
17 0017 1 *
18 0018 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
19 0019 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
20 0020 1 * CORPORATION.
21 0021 1 *
22 0022 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
23 0023 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
24 0024 1 *
25 0025 1 *
26 0026 1 *****
27 0027 1
28 0028 1 ++
29 0029 1 Facility: Command Definition Utility, Table Generator Module 3
30 0030 1
31 0031 1 Abstract: This module is one of a few modules that is responsible
32 0032 1 for generating the blocks that make up the DCL tables.
33 0033 1 The blocks are generated by traversing the intermediate
34 0034 1 representation of the CLD file created by the parsing
35 0035 1 modules.
36 0036 1
37 0037 1 It is recommended that you read over the CLITABDEF.SDL file
38 0038 1 before reading this code.
39 0039 1
40 0040 1 Environment: Standard CDU environment.
41 0041 1
42 0042 1 Author: Paul C. Anagnostopoulos
43 0043 1 Creation: 11 January 1983
44 0044 1
45 0045 1 Modifications:
46 0046 1
47 0047 1 V04-002 PCG0001 Peter George 06-Dec-1983
48 0048 1 Add NEG operator.
49 0049 1
50 0050 1 V04-001 PCA1025 Paul C. Anagnostopoulos 25-Jul-1983
51 0051 1 Only qualifiers should be NEGATABLE by default.
52 0052 1 Change the way in which the CONCAT attribute is determined.
53 0053 1 --
54 0054 1
55 0055 1
56 0056 1 Library 'sys$library:lib';
57 0057 1 require 'clitabdef';
```

GENCODE3
V04-000

N 11
15-Sep-1984 23:38:52
14-Sep-1984 11:58:21

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[CDU.SRC]GENCODE3.B32;1 Page 2
(1)

; 58

0382 1 require 'cdureq';

60	0796	1	!	T A B L E O F C O N T E N T S
61	0797	1	!	-----
62	0798	1		
63	0799	1		forward routine
64	0800	1		cdu\$generate_entity: novalue,
65	0801	1		cdu\$generate_expression: novalue;
66	0802	1		
67	0803	1	!	E X T E R N A L R E F E R E N C E S
68	0804	1	!	-----
69	0805	1		
70	0806	1		external routine
71	0807	1		cdu\$generate_path,
72	0808	1		cdu\$lookup_child,
73	0809	1		cdu\$remember_reference,
74	0810	1		cdu\$report_semantic_error,
75	0811	1		lib\$get_vm;
76	0812	1		
77	0813	1		external
78	0814	1		cdu\$gl_root_node: ref node,
79	0815	1		cdu\$gl_table: pointer;

```

81      0816 1  |++
82      0817 1  |Description: This routine is called to generate an entity block for
83      0818 1  |a parameter, qualifier, or type keyword. The block
84      0819 1  |describes the entity in full detail.
85      0820 1  |
86      0821 1  |Parameters: top_node      By reference, the top-level node
87      0822 1  |                      representing the entity.
88      0823 1  |                      number      The sequential number of the entity.
89      0824 1  |
90      0825 1  |Returns:   Nothing.
91      0826 1  |
92      0827 1  |Notes:
93      0828 1  |--
94      0829 1  |
95      0830 1  |GLOBAL ROUTINE cdu$generate_entity(top_node: ref node,
96      0831 1  |                                number: long)      : novalue
97      0832 2  |= BEGIN
98      0833 2  |
99      0834 2  |local
100     0835 2  |    status: long,
101     0836 2  |    entity: pointer,
102     0837 2  |    variable_ptr: long,
103     0838 2  |    child: ref node,
104     0839 2  |    grandchild: ref node,
105     0840 2  |    definition: ref node,
106     0841 2  |    concatenate_seen: boolean initial(false);
107     0842 2  |
108     0843 2  |
109     0844 2  |! Allocate space for the largest possible entity block.
110     0845 2  |
111     P 0846 2  |allocate_largest_table_block(ent_k_length + ent_k_max_name + ent_k_max_label +
112     0847 2  |                                ent_k_max_prompt + ent_k_max_defval, entity);
113     0848 2  |
114     0849 2  |! Begin by initializing the entity block. This includes any fields that
115     0850 2  |! don't depend on the intermediate representation.
116     0851 2  |
117     0852 2  |entity[ent_b_type] = block_k_entity;
118     0853 3  |entity[ent_b_subtype] = (select neu .top_node[node_w_type] of set
119     0854 3  |                        [node_k_parameter]:   ent_k_parameter;
120     0855 3  |                        [node_k_qualifier]:   ent_k_qualifier;
121     0856 3  |                        [node_k_keyword]:   ent_k_keyword;
122     0857 2  |                        tes);
123     0858 2  |entity[ent_w_flags] = 0;
124     0859 2  |if .top_node[node_w_type] eqlu node_k_qualifier then
125     0860 2  |    entity[ent_v_neg] = true;
126     0861 2  |entity[ent_w_tro_count] = 3;
127     0862 2  |entity[ent_l_next] = entity[ent_l_syntax] = entity[ent_l_user_type] = 0;
128     0863 2  |entity[ent_b_number] = .number;
129     0864 2  |entity[ent_b_valtype] = ent_k_user_defined;
130     0865 2  |entity[ent_w_defval] = 0;
131     0866 2  |
132     0867 2  |! Set up to add information to the variable portion of the block.
133     0868 2  |! Then add the entity name as an ASCII string.
134     0869 2  |
135     0870 2  |variable_ptr = entity[ent_z_variable];
136     0871 2  |entity[ent_w_name] = .variable_ptr - .entity;
137     0872 2  |ch$move(1+.top_node[node_b_text_length],top_node[node_b_text_length],
```



```
138      .variable_ptr);
139      variable_ptr = .variable_ptr + 1+.top_node[node_b_text_length];
140
141      ! Now we scan the children of the top-level node in order to collect
142      ! the various attributes of the entity and place them in the entity block.
143
144      P 0879      scan_children(top_node,child,
145      P 0880
146      P 0881          ! Case on the type of the child.
147      P 0882
148      P 0883      case .child[node_w_type] from 0 to node_k_max_type of set
149      P 0884      [node_k_batch]:
150      P 0885
151      P 0886          ! The BATCH clause simply sets a flag.
152      P 0887
153      P 0888          entity[ent_v_batch] = true;
154      P 0889
155      P 0890      [node_k_cliflags]:
156      P 0891
157      P 0892          ! For the CLIFLAGS clause, we scan the children, each of
158      P 0893          ! which specifies a flag to be set.
159      P 0894
160      P 0895          scan_children(child,grandchild,
161      P 0896
162      P 0897              selectoneu .grandchild[node_w_type] of set
163      P 0898              [node_k_mcrignore]:      entity[ent_v_mcrignore] = true;
164      P 0899              [node_k_mcroptdelim]:    entity[ent_v_mcroptdelim] = true;
165      P 0900              [otherwise]:          cdu$report_semantic_error(msg(cdu$_igncliflag),1,
166      P 0901                                  .grandchild[node_w_line]);
167      P 0902              tes;
168      P 0903          );
169      P 0904
170      P 0905      [node_k_default]:
171      P 0906
172      P 0907          ! The DEFAULT clause simply sets a flag.
173      P 0908
174      P 0909          entity[ent_v_deftrue] = true;
175      P 0910
176      P 0911      [node_k_label]:
177      P 0912
178      P 0913          ! The LABEL clause specifies a symbol by which this entity
179      P 0914          ! is to be retrieved. Add it to the variable portion of
180      P 0915          ! the block as an ASCII string.
181      P 0916
182      P 0917          (entity[ent_w_label] = .variable_ptr - .entity;
183      P 0918          ch$move(1+.child[node_b_text_length],child[node_b_text_length],
184      P 0919              .variable_ptr);
185      P 0920          variable_ptr = .variable_ptr + 1+.child[node_b_text_length];);
186      P 0921
187      P 0922      [node_k_negatable]:
188      P 0923
189      P 0924          ! The NEGATABLE clause simply sets a flag.
190      P 0925
191      P 0926          entity[ent_v_neg] = true;
192      P 0927
193      P 0928      [node_k_nonnegatable]:
194      P 0929
```

```
: 195 P 0930 2 ! The NONNEGATABLE clause simply clears a flag.
: 196 P 0931 2
: 197 P 0932 2 entity[ent_v_neg] = false;
: 198 P 0933 2
: 199 P 0934 2 [node_k_placement]:
: 200 P 0935 2
: 201 P 0936 2 ! The PLACEMENT clause specifies where the entity can
: 202 P 0937 2 ! appear on the command line. The node contains one of
: 203 P 0938 2 ! three symbols, which have already been checked. We can
: 204 P 0939 2 ! just select on the first letter of the symbol.
: 205 P 0940 2
: 206 P 0941 2 selectoneu ch$rchar(child[node_t_text]) of set
: 207 P 0942 2 ['G']: (entity[ent_v_verb] = true;
: 208 P 0943 2 entity[ent_v_parm] = false;);
: 209 P 0944 2 ['L']: (entity[ent_v_verb] = false;
: 210 P 0945 2 entity[ent_v_parm] = true;);
: 211 P 0946 2 ['P']: entity[ent_v_verb] = entity[ent_v_parm] = true;
: 212 P 0947 2 tes;
: 213 P 0948 2
: 214 P 0949 2 [node_k_prompt]:
: 215 P 0950 2
: 216 P 0951 2 ! The PROMPT clause specifies a string which is to be used
: 217 P 0952 2 ! to prompt for a parameter. It is stored in the variable
: 218 P 0953 2 ! portion of the block as an ASCII string.
: 219 P 0954 2
: 220 P 0955 2 (entity[ent_w_prompt] = .variable_ptr - .entity;
: 221 P 0956 2 ch$move(1+.child[node_b_text_length],child[node_b_text_length],
: 222 P 0957 2 .variable_ptr);
: 223 P 0958 2 variable_ptr = .variable_ptr + 1+.child[node_b_text_length];);
: 224 P 0959 2
: 225 P 0960 2 [node_k_syntax]:
: 226 P 0961 2
: 227 P 0962 2 ! The SYNTAX clause specifies the name of a syntax
: 228 P 0963 2 ! definition which is to be used to parse the rest of the
: 229 P 0964 2 ! command. Find the syntax definition.
: 230 P 0965 2
: 231 P 0966 2 (definition = cdu$lookup_child(.cdu$gl_root_node,node_k_define_syntax,
: 232 P 0967 2 .child[node_b_text_length],child[node_t_text]);
: 233 P 0968 2
: 234 P 0969 2 ! Call a routine to remember this reference for later
: 235 P 0970 2 ! resolution. Or perhaps we didn't find it.
: 236 P 0971 2
: 237 P 0972 2 if .definition neq 0 then
: 238 P 0973 2 cdu$remember_reference(entity[ent_l_syntax],.definition)
: 239 P 0974 2 else
: 240 P 0975 2 cdu$report_semantic_error(msg(cdu$undefsyntax),2,
: 241 P 0976 2 .child[node_w_line],child[node_b_text_length]););
: 242 P 0977 2
: 243 P 0978 2 [node_k_value]:
: 244 P 0979 2
: 245 P 0980 2 ! The VALUE clause specifies a set of subclauses which
: 246 P 0981 2 ! define the kind of value this entity can take. Set
: 247 P 0982 2 ! the flag that says the entity can have a value.
: 248 P 0983 2
: 249 P 0984 2 (entity[ent_v_val] = true;
: 250 P 0985 2
: 251 P 0986 2 ! Scan the nodes that represent the subclauses.
```



```
252 P 0987 2
253 P 0988 2
254 P 0989 2
255 P 0990 2
256 P 0991 2
257 P 0992 2
258 P 0993 2
259 P 0994 2
260 P 0995 2
261 P 0996 2
262 P 0997 2
263 P 0998 2
264 P 0999 2
265 P 1000 2
266 P 1001 2
267 P 1002 2
268 P 1003 2
269 P 1004 2
270 P 1005 2
271 P 1006 2
272 P 1007 2
273 P 1008 2
274 P 1009 2
275 P 1010 2
276 P 1011 2
277 P 1012 2
278 P 1013 2
279 P 1014 2
280 P 1015 2
281 P 1016 2
282 P 1017 2
283 P 1018 2
284 P 1019 2
285 P 1020 2
286 P 1021 2
287 P 1022 2
288 P 1023 2
289 P 1024 2
290 P 1025 2
291 P 1026 2
292 P 1027 2
293 P 1028 2
294 P 1029 2
295 P 1030 2
296 P 1031 2
297 P 1032 2
298 P 1033 2
299 P 1034 2
300 P 1035 2
301 P 1036 2
302 P 1037 2
303 P 1038 2
304 P 1039 2
305 P 1040 2
306 P 1041 2
307 P 1042 2
308 P 1043 2
```

```
scan_children(child,grandchild,
! Determine what to do from the node type.
selectoneu .grandchild[node_w_type] of set
[node_k_concatenate]:
! The CONCATENATE clause simply sets a bit.
! Remember that we've seen such a clause.
(entity[ent_v_concat] = true;
concatenate_seen = true;);
[node_k_noconcatenate]:
! And NOCONCATENATE clears it.
! Remember that we've seen such a clause.
(entity[ent_v_concat] = false;
concatenate_seen = true;);
[node_k_default]:
! The DEFAULT clause specifies default
! values which are placed in the variable
! portion of the block as an ASCII string.
(entity[ent_w_defval] = .variable_ptr - .entity;
ch$move(1+.grandchild[node_b_text_length],grandchild[node_b_text_length],
.variable_ptr);
variable_ptr = .variable_ptr + 1+.grandchild[node_b_text_length];);
[node_k_impcat]:
! The IMPCAT clause simply sets a flag.
entity[ent_v_impcat] = true;
[node_k_list]:
! The LIST clause simply sets a flag.
entity[ent_v_list] = true;
[node_k_required]:
! The REQUIRED clause simply sets a flag.
entity[ent_v_valreq] = true;
[node_k_type_builtin]:
! The TYPE clause with a builtin type name
! specifies the class of values this entity
! can have. The type name has already been
```

```
309 P 1044 2
310 P 1045 2
311 P 1046 2
312 P 1047 2
313 P 1048 2
314 P 1049 2
315 P 1050 2
316 P 1051 2
317 P 1052 2
318 P 1053 2
319 P 1054 2
320 P 1055 2
321 P 1056 2
322 P 1057 2
323 P 1058 2
324 P 1059 2
325 P 1060 2
326 P 1061 2
327 P 1062 2
328 P 1063 2
329 P 1064 2
330 P 1065 2
331 P 1066 2
332 P 1067 2
333 P 1068 2
334 P 1069 2
335 P 1070 2
336 P 1071 2
337 P 1072 2
338 P 1073 2
339 P 1074 2
340 P 1075 2
341 P 1076 2
342 P 1077 2
343 P 1078 2
344 P 1079 2
345 P 1080 2
346 P 1081 2
347 P 1082 2
348 1083 2 );
```

```
! translated to its numeric code, so just
! move it into the block.

entity[ent_b_valtype] = ch$rchar(grandchild[node_t_text]);

[node_k_type_user]:
! The TYPE clause with a user-defined type
! name references a type definition. Find
! the definition.
(definition = cdu$lookup_child(.cdu$gl_root_node,node_k_define_type,
    .grandchild[node_b_text_length],grandchild[node_t_text]);

! Call a routine to remember this reference
! for later resolution. Or perhaps we didn't
! find it.
if .definition neq 0 then
    cdu$remember_reference(entity[ent_l_user_type],.definition)
else
    cdu$report_semantic_error(msg(cdu$_undeftype),2,
        .grandchild[node_w_line],grandchild[node_b_text_length]););

[otherwise]:
! Oops, we have some kind of internal error.
signal(msg(cdu$_intinvnode));

tes;

););

[inrange,
outrange]:
! Oops, we have some kind of internal error.
signal(msg(cdu$_intinvnode));

tes;
```



```

350 1084 2 ! Once we have processed all of the clauses, we need to handle additional
351 1085 2 cases which are implied by the clauses.
352 1086 2
353 1087 2     If no [NO]CONCATENATE clause specified, assume same as LIST.
354 1088 2     For a qualifier, if no PLACEMENT specified assume GLOBAL.
355 1089 2     If no LABEL specified, use entity name.
356 1090 2     For a parameter, If no PROMPT specified, use LABEL.
357 1091 2
358 1092 2 if not .concatenate_seen then
359 1093 2     entity[ent_v_concat] = .entity[ent_v_list];
360 1094 2 if .top_node[node_w_type] eqv node_k_qualifier and
361 1095 2     not .entity[ent_v_verb] and not .entity[ent_v_parm] then (
362 1096 2     entity[ent_v_verb] = true;
363 1097 2     entity[ent_v_parm] = false;
364 1098 2 );
365 1099 2 if .entity[ent_w_label] eqv 0 then
366 1100 2     entity[ent_w_label] = .entity[ent_w_name];
367 1101 2 if .top_node[node_w_type] eqv node_k_parameter and
368 1102 2     .entity[ent_w_prompt] eqv 0 then
369 1103 2     entity[ent_w_prompt] = .entity[ent_w_label];
370 1104 2
371 1105 2 ! Set the final length of the entity block.
372 1106 2
373 1107 2 set_table_block_size(.variable_ptr - .entity, entity);
374 1108 2
375 1109 2 ! Place the TRD of the new block in its top-level representation node.
376 1110 2
377 1111 2 top_node[node_l_code] = .entity - .cdu$gl_table;
378 1112 2 return;
379 1113 2
380 1114 1 END;
```

```
.TITLE GENCODE3
.IDENT \V04-000\
```

```
.EXTRN CDUS$GENERATE_PATH
.EXTRN CDUS$LOOKUP_CHILD
.EXTRN CDUS$REMEMBER_REFERENCE
.EXTRN CDUS$REPORT_SEMANTIC_ERROR
.EXTRN LIB$GET_VM, CDUS$GL_ROOT_NODE
.EXTRN CDUS$GL_TABLE, CDUS$_IGNCCIFLAG
.EXTRN CDUS$_UNDEFSYNTAX
.EXTRN CDUS$_UNDEFTYPE, CDUS$_INTINVNODE
```

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

```
.ENTRY CDUS$GENERATE_ENTITY, Save R2,R3,R4,R5,R6,-
```

```

SUBL2 #20, SP
CLRB CONCATENATE_SEEN
PUSHAB ENTITY
MOVZBL #222, 16(SP)
PUSHAB 16(SP)
CALLS #2, LIB$GET_VM
BLBS STATUS, 1$
PUSHL STATUS
```

```

                                OFFC 00000
                                SE      14 C2 00002
                                6E 94 00005
                                AE 9F 00007
                                10 DE 8F 9A 0000A
                                10 AE 9F 0000F
                                00 02 FB 00012
                                09 50 E8 00019
                                50 DD 0001C
```

0830

0832

0847

[illegible]

		00000000G	00	00000000G	8F	DD	00157	PUSHL	#CDUS_IGNCLIFLAG	
		00000000G	00		03	FB	0015D	CALLS	#3, CDUSREPORT_SEMANTIC_ERROR	
			57	04	A7	D0	00164	16\$:	MOVL	4(GRANDCHILD), GRANDCHILD
					CF	11	00168	BRB	13\$	
			6B		04	88	0016A	17\$:	BISB2	#4, (R11)
					58	11	0016D	BRB	27\$	
18	A9		56		59	A3	0016F	18\$:	SUBW3	R9, VARIABLE_PTR, 24(R9)
					3F	11	00174	BRB	26\$	
			6B		02	88	00176	19\$:	BISB2	#2, (R11)
					77	11	00179	BRB	29\$	
			6B		02	8A	0017B	20\$:	BICB2	#2, (R11)
					72	11	0017E	BRB	29\$	
			50	11	A8	9A	00180	21\$:	MOVZBL	17(CHILD), R0
		47	8F		50	91	00184	CMPB	R0, #71	
					0A	12	00188	BNEQ	22\$	
		01	AB		01	88	0018A	BISB2	#1, 1(R11)	
		01	AB		02	8A	0018E	BICB2	#2, 1(R11)	
					76	11	00192	BRB	31\$	
		4C	8F		50	91	00194	22\$:	CMPB	R0, #76
					0A	12	00198	BNEQ	23\$	
		01	AB		01	8A	0019A	BICB2	#1, 1(R11)	
		01	AB		02	88	0019E	BISB2	#2, 1(R11)	
					66	11	001A2	BRB	31\$	
		50	8F		50	91	001A4	23\$:	CMPB	R0, #80
					60	12	001A8	BNEQ	31\$	
		01	AB		03	88	001AA	BISB2	#3, 1(R11)	
					5A	11	001AE	BRB	31\$	
1A	A9		56		59	A3	001B0	24\$:	SUBW3	R9, VARIABLE_PTR, 26(R9)
			5A	10	A8	9A	001B5	25\$:	MOVZBL	16(CHILD), RTO
			50	01	AA	9E	001B9	26\$:	MOVAB	1(R10), R0
			A8		50	28	001BD	MOVAB	R0, 16(CHILD), (VARIABLE_PTR)	
66	10		56	01	AA	46	9E	001C2	MOVAB	1(R10)[VARIABLE_PTR], VARIABLE_PTR
					41	11	001C7	27\$:	BRB	31\$
				11	A8	9F	001C9	28\$:	PUSHAB	17(CHILD)
			7E	10	A8	9A	001CC	MOVZBL	16(CHILD), -(SP)	
					05	DD	001D0	PUSHL	#5	
		00000000G	00	00000000G	00	DD	001D2	PUSHL	CDUSGL_ROOT_NODE	
			04		04	FB	001D8	CALLS	#4, CDUSLOOKUP_CHILD	
			AE		50	D0	001DF	MOVL	R0, DEFINITION	
					0F	13	001E3	BEQL	30\$	
				04	AE	DD	001E5	PUSHL	DEFINITION	
				0C	A9	9F	001E8	PUSHAB	12(R9)	
		00000000G	00		02	FB	001EB	CALLS	#2, CDUSREMEMBER_REFERENCE	
					16	11	001F2	29\$:	BRB	31\$
				10	A8	9F	001F4	30\$:	PUSHAB	16(CHILD)
			7E	02	A8	3C	001F7	MOVZWL	2(CHILD), -(SP)	
					02	DD	001FB	PUSHL	#2	
		00000000G	00	00000000G	8F	DD	001FD	PUSHL	#CDUS_UNDEFSYNTAX	
					04	FB	00203	CALLS	#4, CDUSREPORT_SEMANTIC_ERROR	
			6B		00C7	31	0020A	31\$:	BRW	47\$
			57		01	88	0020D	32\$:	BISB2	#1, (R11)
				08	A8	D0	00210	33\$:	MOVL	8(CHILD), GRANDCHILD
			32		F4	13	00214	BEQL	31\$	
					67	B1	00216	CMPW	(GRANDCHILD), #50	
					06	12	00219	BNEQ	34\$	
			6B	40	8F	88	0021B	BISB2	#64, (R11)	
					09	11	0021F	BRB	35\$	

33		67	B1	00221	34\$:	CMPW	(GRANDCHILD), #51	:	
		09	12	00224		BNEQ	36\$	:	
6B	40	8F	8A	00226		BICB2	#64, (R11)	:	
6E		01	90	0022A	35\$:	MOVB	#1, CONCATENATE_SEEN	:	
		77	11	0022D		BRB	42\$	:	
19		67	B1	0022F	36\$:	CMPW	(GRANDCHILD), #25	:	
		19	12	00232		BNEQ	37\$	:	
1C	A9	56	59	A3	00234	SUBW3	R9, VARIABLE_PTR, 28(R9)	:	
		5A	10	A7	9A	00239	MOVZBL	16(GRANDCHILD), R10	:
		50	01	AA	9E	0023D	MOVAB	1(R10), R0	:
66	10	A7	50	28	00241	MOVC3	R0, 16(GRANDCHILD), (VARIABLE_PTR)	:	
		56	01	AA46	9E	00246	MOVAB	1(R10)[VARIABLE_PTR], VARIABLE_PTR	:
			71	11	0024B	BRB	44\$	:	
1D		67	B1	0024D	37\$:	CMPW	(GRANDCHILD), #29	:	
		06	12	00250		BNEQ	38\$	:	
6B	80	8F	88	00252		BISB2	#128, (R11)	:	
		75	11	00256		BRB	46\$	:	
1E		67	B1	00258	38\$:	CMPW	(GRANDCHILD), #30	:	
		05	12	0025B		BNEQ	39\$	:	
6B		20	88	0025D		BISB2	#32, (R11)	:	
		6B	11	00260		BRB	46\$	:	
1F		67	B1	00262	39\$:	CMPW	(GRANDCHILD), #31	:	
		05	12	00265		BNEQ	40\$	:	
6B		10	88	00267		BISB2	#16, (R11)	:	
		61	11	0026A		BRB	46\$	:	
20		67	B1	0026C	40\$:	CMPW	(GRANDCHILD), #32	:	
		07	12	0026F		BNEQ	41\$	:	
15	A9	11	A7	90	00271	MOVB	17(GRANDCHILD), 21(R9)	:	
			55	11	00276	BRB	46\$	:	
21			67	B1	00278	41\$:	CMPW	(GRANDCHILD), #33	:
			43	12	0027B		BNEQ	45\$	:
		11	A7	9F	0027D	PUSHAB	17(GRANDCHILD)	:	
7E	10	A7	9A	00280		MOVZBL	16(GRANDCHILD), -(SP)	:	
		06	DD	00284		PUSHL	#6	:	
00000000G	00	00	DD	00286		PUSHL	CDU\$GL_ROOT_NODE	:	
04	AE	04	FB	0028C		CALLS	#4, CDU\$LOOKUP_CHILD	:	
		50	D0	00293		MOVL	R0, DEFINITION	:	
		0F	13	00297		BEQL	43\$	:	
		04	AE	DD	00299	PUSHL	DEFINITION	:	
		10	A9	9F	0029C	PUSHAB	16(R9)	:	
00000000G	00	02	FB	0029F		CALLS	#2, CDU\$REMEMBER_REFERENCE	:	
		25	11	002A6	42\$:	BRB	46\$	:	
		10	A7	9F	002A8	43\$:	PUSHAB	16(GRANDCHILD)	:
7E	02	A7	3C	002AB		MOVZWL	2(GRANDCHILD), -(SP)	:	
		02	DD	002AF		PUSHL	#2	:	
00000000G	00	8F	DD	002B1		PUSHL	#CDU\$ UNDEFTYPE	:	
		04	FB	002B7		CALLS	#4, CDU\$REPORT_SEMANTIC_ERROR	:	
		0D	11	002BE	44\$:	BRB	46\$	:	
00000000G	00	8F	DD	002C0	45\$:	PUSHL	#CDU\$ INTINVNODE	:	
		01	FB	002C6		CALLS	#1, LIB\$SIGNAL	:	
57	04	A7	D0	002CD	46\$:	MOVL	4(GRANDCHILD), GRANDCHILD	:	
		FF40	31	002D1		BRW	33\$	:	
58	04	A8	D0	002D4	47\$:	MOVL	4(CHILD), CHILD	:	
		FDD1	31	002D8		BRW	7\$	:	
0A		6E	E8	002DB	48\$:	BLBS	CONCATENATE SEEN, 49\$	:	
01		05	EF	002DE		EXTZV	#5, #1, (R1T), R0	:	
6B	01	50	F0	002E3		INSV	R0, #6, #1, (R11)	:	

		10	0C	AE	E9	002E8	49\$:	BLBC	12(SP), 50\$	:	1094
		0C	01	AB	E8	002EC		BLBS	1(R11), 50\$	:	1095
08		6B		09	E0	002F0		BBS	#9, (R11), 50\$	:	
	01	AB		01	88	002F4		BISB2	#1, 1(R11)	:	1096
	01	AB		02	8A	002F8		BICB2	#2, 1(R11)	:	1097
			18	A9	B5	002FC	50\$:	TSTW	24(R9)	:	1099
				05	12	002FF		BNEQ	51\$	:	
	18	A9	16	A9	B0	00301		MOVW	22(R9), 24(R9)	:	1100
		0D	08	AE	B1	00306	51\$:	CMPW	8(SP), #13	:	1101
				0A	12	0030A		BNEQ	52\$	:	
			1A	A9	B5	0030C		TSTW	26(R9)	:	1102
				05	12	0030F		BNEQ	52\$	:	
	1A	A9	18	A9	B0	00311		MOVW	24(R9), 26(R9)	:	1103
		56		59	C2	00316	52\$:	SUBL2	R9, R6	:	1107
		56		03	C0	00319		ADDL2	#3, R6	:	
		56		04	C6	0031C		DIVL2	#4, R6	:	
69		56		04	A5	0031F		MULW3	#4, R6, (R9)	:	
		50	00000000G	00	D0	00323		MOVL	CDU\$GL TABLE, R0	:	
		51		69	3C	0032A		MOVZWL	(R9), R1	:	
	10	A0		51	C0	0032D		ADDL2	R1, 16(R0)	:	
51	04	AC		0C	C1	00331		ADDL3	#12, TOP_NODE, R1	:	1111
61		59		50	C3	00336		SUBL3	R0, R9, 7(R1)	:	
				04	0033A			RET		:	1114

; Routine Size: 827 bytes, Routine Base: \$CODE\$ + 0000


```
382 1115 1 !++
383 1116 1 !Description: This routine is called to generate an expression block
384 1117 1 !for a boolean expression. If the expression contains an
385 1118 1 !operator, then the expression block specifies the operator
386 1119 1 !and its operands. If the expression is simply a path, then
387 1120 1 !a path expression block is constructed.
388 1121 1
389 1122 1 !Parameters: definition By reference, the node that represents the
390 1123 1 !verb or syntax change definition in which
391 1124 1 !the boolean expression appears.
392 1125 1 !top_node By reference, the top-level node
393 1126 1 !representing the expression. The TRO of
394 1127 1 !the generated expression block will be
395 1128 1 !stored therein.
396 1129 1
397 1130 1 !Returns: Nothing.
398 1131 1
399 1132 1 !Notes:
400 1133 1 !--
401 1134 1
402 1135 1 GLOBAL ROUTINE cdu$generate_expression(definition: ref node,
403 1136 1 !top_node: ref node): novalue
404 1137 2 = BEGIN
405 1138 2
406 1139 2 local
407 1140 2 status: long,
408 1141 2 operand_count: long initial(0),
409 1142 2 expression: pointer,
410 1143 2 child: ref node,
411 1144 2 tro: long;
412 1145 2
413 1146 2
414 1147 2 ! If the expression is simply a path, then call a routine to generate the
415 1148 2 ! path expression block. That's all we have to do.
416 1149 2
417 1150 3 if .top_node[node_w_type] eql node_k_path then (
418 1151 3 cdu$generate_path(.definition,.top_node);
419 1152 3 return;
420 1153 2 );
421 1154 2
422 1155 2 ! The top-level expression node represents an operator. Count the number of
423 1156 2 ! children of the node in order to determine how many operands there are.
424 1157 2
425 1158 2 P scan_children(top_node,child,
426 1159 2 P increment(operand_count);
427 1160 2 );
428 1161 2
429 1162 2 ! Allocate space for an expression block with the appropriate number of
430 1163 2 ! operands.
431 1164 2
432 1165 2 allocate_largest_table_block(exp_k_length + .operand_count*4, expression);
433 1166 2
434 1167 2 ! Initialize the header of the expression block.
435 1168 2
436 1169 2 expression[exp_b_type] = block_k_expression;
437 1170 3 expression[exp_b_subtype] = {select oneu .top_node[node_w_type] of set
438 1171 3 [node_k_not]: exp_k_not;
```



```

: 439      1172      3      [node_k_any2]: exp_k_any2;
: 440      1173      3      [node_k_and]: exp_k_and;
: 441      1174      3      [node_k_or]: exp_k_or;
: 442      1175      3      [node_k_neg]: exp_k_neg;
: 443      1176      2      tes);
: 444      1177      2      expression[exp_w_flags] = 0;
: 445      1178      2      expression[exp_w_tro_count] = 0;
: 446      1179      2
: 447      1180      2      ! Now we scan all of the children again, this time generating expression
: 448      1181      2      ! blocks for each of them. The TROs of these expression blocks are stored
: 449      1182      2      ! as the operands of the top-level operator.
: 450      1183      2
: 451      1184      3      begin
: 452      1185      3      bind
: 453      1186      3      operand_list = expression[exp_l_operand_list]: vector[,long];
: 454      1187      3
: 455      P 1188      3      scan_children(top_node,child,
: 456      P 1189      3
: 457      P 1190      3      ! Generate an expression block for the operand and then store its
: 458      P 1191      3      ! TRO in the operand list of the top-level operator.
: 459      P 1192      3
: 460      P 1193      3      cdu$generate_expression(.definition,.child);
: 461      P 1194      3      operand_list[expression[exp_w_tro_count]] = .child[node_l_code];
: 462      P 1195      3      increment(expression[exp_w_tro_count]);
: 463      1196      3      );
: 464      1197      2      end;
: 465      1198      2
: 466      1199      2      ! Set the size of the expression block in its header.
: 467      1200      2
: 468      1201      2      set_table_block_size(exp_k_length + .operand_count*4, expression);
: 469      1202      2
: 470      1203      2      ! Store the TRO of the new block in its representing node.
: 471      1204      2
: 472      1205      2      top_node[node_l_code] = .expression - .cdu$gl_table;
: 473      1206      2      return;
: 474      1207      2
: 475      1208      1      END;
```

			003C 00000	.ENTRY	CDU\$GENERATE_EXPRESSION, Save R2,R3,R4,R5	: 1135
5E		08	C2 00002	SUBL2	#8, SP	
		52	D4 00005	CLRL	OPERAND COUNT	: 1137
54	08	AC	D0 00007	MOVL	TOP_NODE, R4	: 1150
2E		64	B1 0000B	CMPW	(R4), #46	
		0D	12 0000E	BNEQ	1\$	
		54	DD 00010	PUSHL	R4	: 1151
	04	AC	DD 00012	PUSHL	DEFINITION	
00000000G	00	02	FB 00015	CALLS	#2, CDU\$GENERATE_PATH	
			04 0001C	RET		: 1150
55	08	A4	D0 0001D	MOVL	8(R4), CHILD	: 1160
		08	13 00021	BEQL	3\$	
		52	D6 00023	INCL	OPERAND COUNT	
55	04	A5	D0 00025	MOVL	4(CHILD), CHILD	
		F6	11 00029	BRB	2\$	

04	AE	04	52	04	AE	9F	0002B	3\$:	PUSHAB	EXPRESSION	:	1165	
			AE		02	78	0002E		ASHL	#2, OPERAND_COUNT, 4(SP)	:		
		04			08	C0	00033		ADDL2	#8, 4(SP)	:		
			00000000G	00	04	AE	9F	00037	PUSHAB	4(SP)	:		
				09		02	FB	0003A	CALLS	#2, LIB\$GET_VM	:		
						50	E8	00041	BLBS	STATUS, 4\$	:		
			00000000G	00		50	DD	00044	PUSHL	STATUS	:		
				53	04	01	FB	00046	CALLS	#1, LIB\$SIGNAL	:		
		02	A3			AE	D0	0004D	4\$:	MOVL	EXPRESSION, R3	:	1169
			2C			05	90	00051	MOVB	#5, 2(R3)	:		
						64	B1	00055	CMPW	(R4), #44	:	1171	
				50		05	12	00058	BNEQ	5\$	:		
						02	D0	0005A	MOVL	#2, R0	:		
				2D		2B	11	0005D	BRB	10\$	:		
						64	B1	0005F	5\$:	CMPW	(R4), #45	:	1172
				50		05	12	00062	BNEQ	6\$	:		
						03	D0	00064	MOVL	#3, R0	:		
				2B		21	11	00067	BRB	10\$	:		
						64	B1	00069	6\$:	CMPW	(R4), #43	:	1173
				50		05	12	0006C	BNEQ	7\$	:		
						04	D0	0006E	MOVL	#4, R0	:		
				29		17	11	00071	BRB	10\$	:		
						64	B1	00073	7\$:	CMPW	(R4), #41	:	1174
				50		05	12	00076	BNEQ	8\$	:		
						05	D0	00078	MOVL	#5, R0	:		
				35		0D	11	0007B	BRB	10\$	:		
						64	B1	0007D	8\$:	CMPW	(R4), #53	:	1175
				50		05	13	00080	BEQL	9\$	:		
						01	CE	00082	MNEGL	#1, R0	:		
				50		03	11	00085	BRB	10\$	:		
						07	D0	00087	9\$:	MOVL	#7, R0	:	
		03	A3			50	90	0008A	10\$:	MOVB	R0, 3(R3)	:	1170
						A3	D4	0008E		CLRL	4(R3)	:	1177
			55		04	A4	D0	00091		MOVL	8(R4), CHILD	:	1196
					08	1D	13	00095	11\$:	BEQL	12\$	:	
						55	DD	00097		PUSHL	CHILD	:	
					04	AC	DD	00099		PUSHL	DEFINITION	:	
		FF5F	CF			02	FB	0009C		CALLS	#2, CDU\$GENERATE_EXPRESSION	:	
			50		06	A3	3C	000A1		MOVZWL	6(R3), R0	:	
		08	A340		0C	A5	D0	000A5		MOVL	12(CHILD), 8(R3)[R0]	:	
					06	A3	B6	000AB		INCW	6(R3)	:	
			55		04	A5	D0	000AE		MOVL	4(CHILD), CHILD	:	
						E1	11	000B2		BRB	11\$	:	
			52			04	C4	000B4	12\$:	MULL2	#4, R2	:	1201
			52			0B	C0	000B7		ADDL2	#11, R2	:	
			52			04	C6	000BA		DIVL2	#4, R2	:	
63			52			04	A5	000BD		MULW3	#4, R2, (R3)	:	
			50	00000000G		00	D0	000C1		MOVL	CDU\$GL_TABLE, R0	:	
			51			63	3C	000C8		MOVZWL	(R3), R1	:	
		10	A0			51	C0	000CB		ADDL2	R1, 16(R0)	:	
0C	A4		53			50	C3	000CF		SUBL3	R0, R3, 12(R4)	:	1205
						04	000D4			RET		:	1208

; Routine Size: 213 bytes, Routine Base: \$CODE\$ + 033B

; 476 1209 1 END

GENCODE3
V04-000

D 13
15-Sep-1984 23:38:52
14-Sep-1984 11:58:21

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[CDU.SRC]GENCODE3.B32;1
Page 18
(5)

: 477 1210 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

:
: Name Bytes Attributes
: \$CODE\$ 1040 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)

Library Statistics

:
: File ----- Symbols ----- Pages Processing
: Total Loaded Percent Mapped Time
: _\$255\$DUA28:[SYSLIB]LIB.L32;1 18619 4 0 1000 00:01.9

COMMAND QUALIFIERS

:
: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:GENCODE3/OBJ=OBJ\$:GENCODE3 MSRC\$:GENCODE3/UPDATE=(ENH\$:GENCODE3)

: Size: 1040 code + 0 data bytes
: Run Time: 00:25.5
: Elapsed Time: 01:06.2
: Lines/CPU Min: 2851
: Lexemes/CPU-Min: 24617
: Memory Used: 301 pages
: Compilation Complete

0043

AH-BT13A-SE
 VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY