

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          IIIIII          SSSSSSSS
LL          IIIIII          SSSSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SSSSSS
LL          II             SSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LLLLLLLLLLLL IIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS

```



```
1 0001 0 MODULE gencode2 (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 2
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 CLIDEF.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: 12 January 1983
44 0044 1
45 0045 1 Modifications:
46 0046 1 --
47 0047 1
48 0048 1
49 0049 1 library 'sys$library:lib';
50 0050 1 require 'clitabdef';
51 0375 1 require 'cdureq';
```

53	0789	1	!	T A B L E O F C O N T E N T S	
54	0790	1	!	-----	
55	0791	1			
56	0792	1	forward	routine	
57	0793	1		cdu\$generate_command: novalue,	
58	0794	1		cdu\$generate_outputs_list,	
59	0795	1		cdu\$generate_type: novalue;	
60	0796	1			
61	0797	1			
62	0798	1	!	E X T E R N A L R E F E R E N C E S	
63	0799	1	!	-----	
64	0800	1			
65	0801	1	external	routine	
66	0802	1		cdu\$add_verb_name,	
67	0803	1		cdu\$generate_entity,	
68	0804	1		cdu\$generate_expression,	
69	0805	1		cdu\$lookup_child,	
70	0806	1		cdu\$report_semantic_error,	
71	0807	1		cli\$present,	
72	0808	1		lib\$get_vm,	
73	0809	1		lookup_verb_type;	
74	0810	1			
75	0811	1	external		
76	0812	1		cdu\$gl_table: pointer;	
77	0813	1			
78	0814	1	global		
79	0815	1		clitype: byte;	! Temporary hack for ROUTINES.


```

: 81      0816 1  ++
: 82      0817 1  Description: This routine is called to generate a command block which
: 83      0818 1  defines a verb or a syntax change. Additional blocks
: 84      0819 1  may be generated and chained off the command block.
: 85      0820 1
: 86      0821 1  Parameters: top_node      By reference, the node that represents the
: 87      0822 1  verb or syntax change definition.
: 88      0823 1
: 89      0824 1  Returns: Nothing.
: 90      0825 1
: 91      0826 1  Notes:
: 92      0827 1  --
: 93      0828 1
: 94      0829 1  GLOBAL ROUTINE cdu$generate_command(top_node: ref node)      : novalue
: 95      0830 2  = BEGIN
: 96      0831 2
: 97      0832 2  local
: 98      0833 2      status: long,
: 99      0834 2      doing_verb: boolean,
:100     0835 2      command: pointer,
:101     0836 2      verb_name_dsc: descriptor,
:102     0837 2      variable_ptr: pointer,
:103     0838 2      child: ref node,
:104     0839 2      grandchild: ref node,
:105     0840 2      last_parm: pointer,
:106     0841 2      qual_counter: long initial(0),
:107     0842 2      last_qual: pointer,
:108     0843 2      disallow_count: long initial(0),
:109     0844 2      outputs_node: ref node initial(0),
:110     0845 2      expression: ref node;
:111     0846 2
:112     0847 2
:113     0848 2  ! Set a flag to say whether this is a verb definition or not.
:114     0849 2
:115     0850 2  doing_verb = .top_node[node_w_type] eqlu node_k_define_verb;
:116     0851 2
:117     0852 2  ! Allocate enough space to contain the largest possible command block.
:118     0853 2
:119     P 0854 2  allocate_largest_table_block(cmd_k_length + cmd_k_max_name + cmd_k_max_image +
:120     0855 2      cmd_k_max_outputs + cmd_k_max_prefix, command);
:121     0856 2
:122     0857 2  ! Begin by initializing the command block. This includes any fields that
:123     0858 2  ! don't depend on the intermediate representation.
:124     0859 2
:125     0860 2  command[cmd_b_type] = block_k_command;
:126     0861 2  command[cmd_b_subtype] = (if .doing_verb then cmd_k_verb else cmd_k_syntax);
:127     0862 2  command[cmd_w_flags] = 0;
:128     0863 2  command[cmd_w_tro_count] = 3;
:129     0864 2  command[cmd_l_parms] = command[cmd_l_qual] = command[cmd_l_disallow] = 0;
:130     0865 2  command[cmd_b_handler] = cmd_k_none;
:131     0866 2  command[cmd_v_minparm] = command[cmd_v_maxparm] = 0;
:132     0867 2  clitype = (if cli$present(dtext('CLI-MCR')) then vec_k_mcr else vec_k_dcl);
:133     0868 2  build_descriptor(verb_name_dsc, .top_node[node_b_text_length], top_node[node_t_text]);
:134     0869 2  command[cmd_b_verbtyp] = lookup_verb_type(verb_name_dsc);
:135     0870 2  command[cmd_w_name] = command[cmd_w_image] = command[cmd_w_outputs] = command[cmd_w_prefix] = 0;
:136     0871 2
:137     0872 2  ! Set up to add information to the variable portion of the block.
```

```

138      0873 2
139      0874 2 variable_ptr = command[cmd_z_variable];
140      0875 2
141      0876 2 ! Process the verb name(s) or the command name.
142      0877 2
143      0878 2 if .doing_verb then (
144      0879 2     local
145      0880 2         work_ptr: pointer,
146      0881 2
147      0882 2         ! The variable portion of the command block will contain an ASCII
148      0883 2         ! string of ASCII strings, the first of which is the verb name and
149      0884 2         ! the remainder the verb synonyms. Move in the verb name.
150      0885 2
151      0886 2 ch$move(1+.top_node[node_b_text_length],top_node[node_b_text_length],.variable_ptr+1);
152      0887 2 work_ptr = .variable_ptr+1+1+.top_node[node_b_text_length];
153      0888 2
154      0889 2 ! Add the verb name to verb name table.
155      0890 2
156      0891 2 cdu$add_verb_name(verb_name_dsc,.command);
157      0892 2
158      0893 2 ! Scan the children of this definition looking for synonym nodes.
159      0894 2
160      0895 2 scan_children(top_node,child,
161      0896 2
162      0897 2     ! If we have a synonym node, then move in the synonym.
163      0898 2     ! Also add the synonym to the verb name table.
164      0899 2
165      0900 2     if .child[node_w_type] eglu node_k_synonym then (
166      0901 2         ch$move(1+.child[node_b_text_length],child[node_b_text_length],
167      0902 2             .work_ptr);
168      0903 2         work_ptr = .work_ptr + 1+.top_node[node_b_text_length];
169      0904 2         build_descriptor(verb_name_dsc,.child[node_b_text_length],child[node_t_text]);
170      0905 2         cdu$add_verb_name(verb_name_dsc,.command);
171      0906 2
172      0907 2     );
173      0908 2 );
174      0909 2
175      0910 2 ! Set the length of the overall ASCII string.
176      0911 2
177      0912 2 variable_ptr[0,0,8,0] = .work_ptr - .variable_ptr - 1;
178      0913 2
179      0914 2 ) else
180      0915 2
181      0916 2 ! The variable portion of the command block will contain the syntax
182      0917 2 ! name as an ASCII string.
183      0918 2
184      0919 2 ch$move(1+.top_node[node_b_text_length],top_node[node_b_text_length],.variable_ptr);
185      0920 2
186      0921 2 ! Store the BRO of the variable part we just generated, and adjust the
187      0922 2 ! variable portion pointer.
188      0923 2
189      0924 2 command[cmd_w_name] = .variable_ptr - .command;
190      0925 2 variable_ptr = .variable_ptr + 1+.variable_ptr[0,0,8,0];
```



```
: 192      0926 2 ! Now we scan the children of the top-level node in order to collect the
: 193      0927 2 ! various attributes of the command and place them in the command block.
: 194      0928 2
: 195      P 0929 2 scan_children(top_node,child,
: 196      P 0930 2
: 197      P 0931 2     ! Case on the type of the child.
: 198      P 0932 2
: 199      P 0933 2     case .child[node_w_type] from 0 to node_k_max_type of set
: 200      P 0934 2     [node_k_cliflags]:
: 201      P 0935 2
: 202      P 0936 2         ! For the CLIFLAGS clause, we scan the children, each of
: 203      P 0937 2         ! which specifies a flag to be set.
: 204      P 0938 2
: 205      P 0939 2         scan_children(child,grandchild,
: 206      P 0940 2
: 207      P 0941 2             selectoneu .grandchild[node_w_type] of set
: 208      P 0942 2             [node_k_abbrev]:      command[cmd_v_abbrev] = true;
: 209      P 0943 2             [node_k_foreign]:      command[cmd_v_foreign] = true;
: 210      P 0944 2             [node_k_immed]:      command[cmd_v_immed] = true;
: 211      P 0945 2             [node_k_mcrparse]:    command[cmd_v_mcrparse] = true;
: 212      P 0946 2             [node_k_nostat]:     command[cmd_v_nostat] = true;
: 213      P 0947 2             [otherwise]:         cdu$report_semantic_error(msg(cdu$_igncliflag),1,
: 214      P 0948 2                                     .grandchild[node_w_line]);
: 215      P 0949 2             tes;
: 216      P 0950 2         );
: 217      P 0951 2
: 218      P 0952 2     [node_k_cliroutine]:
: 219      P 0953 2
: 220      P 0954 2         ! The CLIROUTINE clause specifies the name of an internal
: 221      P 0955 2         ! CLI routine which is called to perform the command. Set
: 222      P 0956 2         ! the handler code accordingly.
: 223      P 0957 2
: 224      P 0958 2         (command[cmd_b_handler] = cmd_k_cli;
: 225      P 0959 2
: 226      P 0960 2         ! Copy the name of the CLI routine into the variable portion
: 227      P 0961 2         ! of the command block.
: 228      P 0962 2
: 229      P 0963 2         command[cmd_w_image] = .variable_ptr - .command;
: 230      P 0964 2         ch$move(1+.child[node_b_text_length],child[node_b_text_length],
: 231      P 0965 2             .variable_ptr);
: 232      P 0966 2         variable_ptr = .variable_ptr + 1+.child[node_b_text_length];);
: 233      P 0967 2
: 234      P 0968 2     [node_k_disallow]:
: 235      P 0969 2
: 236      P 0970 2         ! A DISALLOW clause specifies a boolean combination of
: 237      P 0971 2         ! entities which are invalid. Generate code for the
: 238      P 0972 2         ! boolean expression.
: 239      P 0973 2
: 240      P 0974 2         (grandchild = .child[node_l_child];
: 241      P 0975 2         cdu$generate_expression(.top_node,.grandchild);
: 242      P 0976 2
: 243      P 0977 2         ! Propagate the TRO of the resulting expression block up
: 244      P 0978 2         ! to the disallow node.
: 245      P 0979 2
: 246      P 0980 2         child[node_l_code] = .grandchild[node_l_code];
: 247      P 0981 2
: 248      P 0982 2         ! Set the flag saying that disallow info has been supplied.
```

```
249 P 0983 2
250 P 0984 2
251 P 0985 2
252 P 0986 2
253 P 0987 2
254 P 0988 2
255 P 0989 2
256 P 0990 2
257 P 0991 2
258 P 0992 2
259 P 0993 2
260 P 0994 2
261 P 0995 2
262 P 0996 2
263 P 0997 2
264 P 0998 2
265 P 0999 2
266 P 1000 2
267 P 1001 2
268 P 1002 2
269 P 1003 2
270 P 1004 2
271 P 1005 2
272 P 1006 2
273 P 1007 2
274 P 1008 2
275 P 1009 2
276 P 1010 2
277 P 1011 2
278 P 1012 2
279 P 1013 2
280 P 1014 2
281 P 1015 2
282 P 1016 2
283 P 1017 2
284 P 1018 2
285 P 1019 2
286 P 1020 2
287 P 1021 2
288 P 1022 2
289 P 1023 2
290 P 1024 2
291 P 1025 2
292 P 1026 2
293 P 1027 2
294 P 1028 2
295 P 1029 2
296 P 1030 2
297 P 1031 2
298 P 1032 2
299 P 1033 2
300 P 1034 2
301 P 1035 2
302 P 1036 2
303 P 1037 2
304 P 1038 2
305 P 1039 2

command[cmd_v_disallows] = true;

! Count the number of DISALLOW statements for use below.
increment(disallow_count);

[node_k_nodisallows]:
! The NODISALLOWS clause specifies that this verb or syntax
! change has no disallow expressions. Set the flag saying
! that the disallow info is relevant. The absence of a
! disallows expression block will tell DCL that there are
! none.
command[cmd_v_disallows] = true;

[node_k_image]:
! The IMAGE clause specifies the file spec of the image to
! be run when the verb is entered. This is treated exactly
! as the CLIRoutine case above, except that the handler code
! is different.
(command[cmd_b_handler] = cmd_k_image;
command[cmd_w_image] = .variable_ptr - .command;
ch$move(1+.child[node_b_text_length],child[node_b_text_length],
.variable_ptr);
variable_ptr = .variable_ptr + 1+.child[node_b_text_length];);

[node_k_outputs]:
! Remember the address of the OUTPUTS node so we can process
! it later.
outputs_node = .child;

[node_k_parameter]:
! We have a PARAMETER clause, which defines a parameter.
! Generate an entity block for it, which will tell us
! if the parameter is required.
(increment(command[cmd_v_maxparm]);
cdu$generate_entity(.child,.command[cmd_v_maxparm]);

! Form the entity blocks into a list, with the TRO of the
! first one in the command block.
if .command[cmd_v_maxparm] eq 1 then
    command[cmd_l_parms] = .child[node_l_code]
else
    last_parm[ent_l_next] = .child[node_l_code];
last_parm = .cdu$gl_table + .child[node_l_code];

! Set the flag saying that parameter info has been supplied.
```



```

: 306      P 1040      2      command[cmd_v_parms] = true;
: 307      P 1041      2
: 308      P 1042      2      ! If the parameter is required, then increment the minimum
: 309      P 1043      2      ! parameter count. Required parameters cannot follow optional
: 310      P 1044      2      ! ones.
: 311      P 1045      2
: 312      P 1046      2      begin
: 313      P 1047      2      bind
: 314      P 1048      2          entity = .cdu$gl_table + .child[node_l_code]: block[,byte];
: 315      P 1049      2
: 316      P 1050      2      if .entity[ent_v_valreq] then (
: 317      P 1051      2          increment(command[cmd_v_minparm]);
: 318      P 1052      2          if .command[cmd_v_maxparm] gtru .command[cmd_v_minparm] then
: 319      P 1053      2              cdu$report_semantic_error(msg(cdu$_invreqparm),1,.child[node_w_line]);
: 320      P 1054      2      );
: 321      P 1055      2      end;);
: 322      P 1056      2
: 323      P 1057      2      [node_k_noparameters]:
: 324      P 1058      2
: 325      P 1059      2      ! The NOPARAMETERS clause specifies that this verb or syntax
: 326      P 1060      2      ! change takes no parameters. Set the flag saying that the
: 327      P 1061      2      ! parameter info is relevent. The absence of a list of
: 328      P 1062      2      ! entity blocks will tell DCL that there are no parameters.
: 329      P 1063      2
: 330      P 1064      2      command[cmd_v_parms] = true;
: 331      P 1065      2
: 332      P 1066      2      [node_k_prefix]:
: 333      P 1067      2
: 334      P 1068      2      ! Save the symbol prefix specified in the PREFIX clause as
: 335      P 1069      2      ! an ASCII string.
: 336      P 1070      2
: 337      P 1071      2      (command[cmd_w_prefix] = .variable_ptr - .command;
: 338      P 1072      2      ch$move(1+.child[node_b_text_length],child[node_b_text_length],
: 339      P 1073      2      .variable_ptr);
: 340      P 1074      2      variable_ptr = .variable_ptr + 1+.child[node_b_text_length];);
: 341      P 1075      2
: 342      P 1076      2      [node_k_qualifier]:
: 343      P 1077      2
: 344      P 1078      2      ! We have a QUALIFIER clause, which defines a qualifier.
: 345      P 1079      2      ! Generate an entity block for it.
: 346      P 1080      2
: 347      P 1081      2      (increment(qual_counter);
: 348      P 1082      2      cdu$generate_entity(.child,.qual_counter);
: 349      P 1083      2
: 350      P 1084      2      ! Form the entity blocks into a list, with the TRO of the
: 351      P 1085      2      ! first one in the command block.
: 352      P 1086      2
: 353      P 1087      2      if .qual_counter eq 1 then
: 354      P 1088      2          command[cmd_l_qual] = .child[node_l_code]
: 355      P 1089      2      else
: 356      P 1090      2          last_qual[ent_l_next] = .child[node_l_code];
: 357      P 1091      2      last_qual = .cdu$gl_table + .child[node_l_code];
: 358      P 1092      2
: 359      P 1093      2      ! Set the flag saying that qualifier info has been supplied.
: 360      P 1094      2
: 361      P 1095      2      command[cmd_v_qual] = true;);
: 362      P 1096      2
```

```
: 363 P 1097 2
: 364 P 1098 2
: 365 P 1099 2
: 366 P 1100 2
: 367 P 1101 2
: 368 P 1102 2
: 369 P 1103 2
: 370 P 1104 2
: 371 P 1105 2
: 372 P 1106 2
: 373 P 1107 2
: 374 P 1108 2
: 375 P 1109 2
: 376 P 1110 2
: 377 P 1111 2
: 378 P 1112 2
: 379 P 1113 2
: 380 P 1114 2
: 381 P 1115 2
: 382 P 1116 2
: 383 P 1117 2
: 384 P 1118 2
: 385 P 1119 2
: 386 P 1120 2
: 387 P 1121 2
: 388 P 1122 2
: 389 P 1123 2
: 390 P 1124 2
: 391 P 1125 2
: 392 P 1126 2
: 393 P 1127 2
: 394 P 1128 2
: 395 P 1129 2
: 396 P 1130 2
: 397 P 1131 2
: 398 P 1132 2
: 399 P 1133 2
: 400 P 1134 2
: 401 P 1135 2
: 402 P 1136 2
: 403 P 1137 2
: 404 P 1138 2
: 405 P 1139 2
: 406 P 1140 2
: 407 1141 2 );
```

[node_k_noqualifiers]:

```
! The NOQUALIFIERS clause specifies that this verb or syntax
! change takes no qualifiers. Set the flag saying that the
! qualifier info is relevant. The absence of a list of
! entity blocks will tell DCL that there are no qualifiers.
```

```
command[cmd_v_qual] = true;
```

[node_k_routine]:

```
! The ROUTINE clause specifies the name of a routine in the
! user's program which is called to perform the command.
! Set the handler type accordingly.
```

```
(command[cmd_b_handler] = cmd_k_user;
```

```
! Allocate a longword in the variable portion of the block
! which will be filled in with the routine address by the
! Linker.
```

```
command[cmd_w_image] = .variable_ptr - .command;
variable_ptr = .variable_ptr + 4;
```

```
! Place the name of the routine after the longword, stored
! as an ASCII string.
```

```
ch$move(1+.child[node_b_text_length],child[node_b_text_length],
        .variable_ptr);
variable_ptr = .variable_ptr + 1+.child[node_b_text_length];);
```

[node_k_synonym]:

```
! These nodes were already processed up above.
```

```
;
```

[inrange,
outrange]:

```
! Oops, we have some kind of internal error.
```

```
signal(msg(cdu$_intinvnode));
```

tes;


```

: 409      1142 2 ! Now we may need to do some additional processing for DISALLOW clauses.
: 410      1143 2 ! This involves creating an expression block which ORs together all of the
: 411      1144 2 ! boolean expressions specified in DISALLOW clauses. We have counted the
: 412      1145 2 ! number of clauses, so allocate space for the expression block.
: 413      1146 2
: 414      1147 2 if .disallow_count nequ 0 then (
: 415      1148 2     allocate_largest_table_block(exp_k_length + .disallow_count*4, expression);
: 416      1149 2
: 417      1150 2     ! Initialize the header of the expression block.
: 418      1151 2
: 419      1152 2     expression[exp_b_type] = block_k_expression;
: 420      1153 2     expression[exp_b_subtype] = exp_k_or;
: 421      1154 2     expression[exp_w_flags] = 0;
: 422      1155 2     expression[exp_w_tro_count] = 0;
: 423      1156 2
: 424      1157 2     ! Find all of the DISALLOW clauses and store the TRO of the
: 425      1158 2     ! corresponding expression blocks as the operands of this OR block.
: 426      1159 2
: 427      1160 2     begin
: 428      1161 2     bind
: 429      1162 2         operand_list = expression[exp_l_operand_list]: vector[,long];
: 430      1163 2
: 431      1164 2     scan_children(top_node,child,
: 432      1165 2         if .child[node_w_type] eqlu node_k_disallow then (
: 433      1166 2             operand_list[expression[exp_w_tro_count]] = .child[node_l_code];
: 434      1167 2             increment(expression[exp_w_tro_count]);
: 435      1168 2         );
: 436      1169 2     );
: 437      1170 2     end;
: 438      1171 2
: 439      1172 2     ! Set the size of the expression block in its header.
: 440      1173 2
: 441      1174 2     set_table_block_size(exp_k_length + .disallow_count*4, expression);
: 442      1175 2
: 443      1176 2     ! Store the TRO of this new OR expression block in the command block.
: 444      1177 2
: 445      1178 2     command[cmd_l_disallow] = .expression - .cdu$gl_table;
: 446      1179 2 );
: 447      1180 2
: 448      1181 2 ! If there was an outputs clause, then we can process it now.
: 449      1182 2
: 450      1183 2 if .outputs_node nequ 0 then (
: 451      1184 2     command[cmd_w_outputs] = .variable_ptr - .command;
: 452      1185 2     variable_ptr = .variable_ptr +
: 453      1186 2         cdu$generate_outputs_list(.top_node, .outputs_node, .variable_ptr);
: 454      1187 2 );
```

```

: 456      1188 2 ! Once we have processed all of the clauses, we need to handle additional
: 457      1189 2 cases which are implied by the clauses.
: 458      1190 2
: 459      1191 2     If no command handler has been specified, then apply some defaults.
: 460      1192 2     Object files can only have user routine handlers, and CLI table
: 461      1193 2     images cannot.
: 462      1194 2
: 463      1195 2 if .command[cmd_b_handler] eqv cmd_k_none then
: 464      1196 3     if .doing_verb then (
: 465      1197 4         if not cli$present(dtext('OBJECT')) then (
: 466      1198 4             command[cmd_b_handler] = cmd_k_image;
: 467      1199 4             command[cmd_w_image] = .command[cmd_w_name] + 1;
: 468      1200 4         )
: 469      1201 2     ) else
: 470      1202 2         command[cmd_b_handler] = cmd_k_same;
: 471      1203 2
: 472      1204 2 if cli$present(dtext('OBJECT')) then (
: 473      1205 3     if .command[cmd_b_handler] eqv cmd_k_cli or
: 474      1206 3         .command[cmd_b_handler] eqv cmd_k_image then
: 475      1207 3         cdu$report_semantic_error(msg(cdu$_routreq),1,.top_node[node_w_line]);
: 476      1208 2 ) else
: 477      1209 2     if .command[cmd_b_handler] eqv cmd_k_user then
: 478      1210 2         cdu$report_semantic_error(msg(cdu$_invrout),1,.top_node[node_w_line]);
: 479      1211 2
: 480      1212 2 ! Set the final size of the command block in its header.
: 481      1213 2
: 482      1214 2 set_table_block_size(.variable_ptr - .command, command);
: 483      1215 2
: 484      1216 2 ! Place the TRO of the new block in its top-level representation node.
: 485      1217 2
: 486      1218 2 top_node[node_l_code] = .command - .cdu$gl_table;
: 487      1219 2 return;
: 488      1220 2
: 489      1221 1 END;
```

```

: INFO#250      L1:1141
: Referenced LOCAL symbol LAST_PARM is probably not initialized
: INFO#250      L1:1141
: Referenced LOCAL symbol LAST_QUAL is probably not initialized
```

```

.TITLE GENCODE2
.IDENT \V04-000\

.PSECT $SPLITS$,NOWRT,NOEXE,2

00 52 43 4D 5F 49 4C 43 00000 P.AAB: .ASCII \CLI MCR\<0>
                                010E0007 00008 P.AAA: .LONG 17694727
                                00000000' 0000C .ADDRESS P.AAB
00 00 54 43 45 4A 42 4F 00010 P.AAD: .ASCII \OBJECT\<0><0>
                                010E0006 00018 P.AAC: .LONG 17694726
                                00000000' 0001C .ADDRESS P.AAD
00 00 54 43 45 4A 42 4F 00020 P.AAF: .ASCII \OBJECT\<0><0>
                                010E0006 00028 P.AAE: .LONG 17694726
                                00000000' 0002C .ADDRESS P.AAF

.PSECT $GLOBALS$,NOEXE,2
```


00000 CLITYPE::

.BLKB 1

.EXTRN CDUSADD VERB_NAME
.EXTRN CDUSGENERATE_ENTITY
.EXTRN CDUSGENERATE_EXPRESSION
.EXTRN CDUSLOOKUP_CHILD
.EXTRN CDUSREPORT_SEMANTIC_ERROR
.EXTRN CLISPRESNT, LIB\$GET_VM
.EXTRN LOOKUP_VERB_TYPE
.EXTRN CDUSGL_TABLE, CDUS_IGNCLIFLAG
.EXTRN CDUS_INVREQPARM
.EXTRN CDUS_INTINVNODE
.EXTRN CDUS_ROUTREQ, CDUS_INVROUT

.PSECT \$CODE\$,NOWRT,2

OFFC 00000

5E	38	C2	00002	.ENTRY	CDUSGENERATE_COMMAND, Save R2,R3,R4,R5,R6,-	0829
	58	D4	00005	SUBL2	R7,R8,R9,R10,R11	
	04	AE	7C 00007	CLRL	#56, SP	0830
5B	04	AC	D0 0000A	CLRQ	DISALLOW COUNT	
		50	D4 0000E	MOV L	OUTPUTS_NODE	0850
04		6B	B1 00010	CLRL	TOP_NODE, R11	
		02	12 00013	CMPL	R0	
		50	D6 00015	BNEQ	(R11), #4	
6E		50	90 00017 1\$:	INCL	1\$	
	28	AE	9F 0001A	MOV B	R0, DOING_VERB	
28	AE	8F	9A 0001D	PUSHAB	COMMAND	0855
	28	AE	9F 00022	MOVZBL	#168, 40(SP)	
00000000G	00		02 FB 00025	PUSHAB	40(SP)	
	09		50 E8 0002C	CALLS	#2, LIB\$GET_VM	
			50 DD 0002F	BLBS	STATUS, 2\$	
00000000G	00		01 FB 00031	PUSHL	STATUS	
	5A	28	AE D0 00038 2\$:	CALLS	#1, LIB\$SIGNAL	0860
02	AA		02 90 0003C	MOV L	COMMAND, R10	
	05		6E E9 00040	MOV B	#2, 2(R10)	0861
	50		01 D0 00043	BLBC	DOING_VERB, 3\$	
			03 11 00046	MOV L	#1, R0	
	50		02 D0 00048 3\$:	BRB	4\$	
03	AA		50 90 0004B 4\$:	MOV L	#2, R0	
14	AE	04	AA 9E 0004F	MOV B	R0, 3(R10)	
		14	BE B4 00054	MOVAB	4(R10), 20(SP)	0862
06	AA		03 B0 00057	CLRW	@20(SP)	
		0C	AA 7C 0005B	MOVW	#3, 6(R10)	0863
		08	AA D4 0005E	CLRQ	12(R10)	0864
		14	AA 9E 00061	CLRL	8(R10)	
24	AE	24	BE 94 00066	MOVAB	20(R10), 36(SP)	0865
		15	AA 9E 00069	CLRB	@36(SP)	
10	AE	F0	8F 8A 0006E	MOVAB	21(R10), 16(SP)	0866
10	BE		0F 8A 00073	BICB2	#240, @16(SP)	
		0000'	CF 9F 00077	BICB2	#15, @16(SP)	
00000000G	00		01 FB 0007B	PUSHAB	P.AAA	0867
	05		50 E9 00082	CALLS	#1, CLISPRESNT	
	50		02 D0 00085	BLBC	R0, 5\$	
			03 11 00088	MOV L	#2, R0	
				BRB	6\$	

		0000'	50	01	D0	0008A	5\$:	MOVL	#1, R0		
		1C	CF	50	90	0008D	6\$:	MOVB	R0, CLITYPE		
		30	AE	10	AB	9A	00092	MOVZBL	16(R11), 28(SP)	0868	
		34	AE	1C	AE	3C	00097	MOVZWL	28(SP), VERB_NAME_DSC		
				11	AB	9E	0009C	MOVAB	17(R11), VERB_NAME_DSC+4		
				30	AE	9F	000A1	PUSHAB	VERB_NAME_DSC	0869	
		00000000G	00	01	FB	000A4		CALLS	#1, LOOKUP VERB_TYPE		
		16	AA	50	90	000AB		MOVB	R0, 22(R10)		
		20	AE	1A	AA	9E	000AF	MOVAB	26(R10), 32(SP)	0870	
				1C	AA	D4	000B4	CLRL	28(R10)		
				20	BE	B4	000B7	CLRW	32(SP)		
				18	AA	B4	000BA	CLRW	24(R10)		
				20	AA	9E	000BD	MOVAB	32(R10), VARIABLE_PTR	0874	
			56	6E	E9	000C1		BLBC	DOING VERB, 10\$	0886	
01	50	1C	AE	01	C1	000C4		ADDL3	#1, 28(SP), R0		
	A6	10	AB	50	28	000C9		MOVC3	R0, 16(R11), 1(VARIABLE_PTR)		
	50	1C	AE	02	C1	000CF		ADDL3	#2, 28(SP), R0	0887	
	57		56	50	C1	000D4		ADDL3	R0, VARIABLE_PTR, WORK_PTR		
				5A	DD	000D8		PUSHL	R10	0891	
				34	AE	9F	000DA	PUSHAB	VERB_NAME_DSC		
		00000000G	00	02	FB	000DD		CALLS	#2, CDU\$ADD VERB_NAME		
			59	08	AB	D0	000E4	MOVL	8(R11), CHILD	0908	
				37	13	000E8	7\$:	BEQL	9\$		
			34	69	B1	000EA		CMPW	(CHILD), #52		
				2C	12	000ED		BNEQ	8\$		
			50	10	A9	9A	000EF	MOVZBL	16(CHILD), R0		
				50	D6	000F3		INCL	R0		
	67	10	A9	50	28	000F5		MOVC3	R0, 16(CHILD), (WORK_PTR)		
	50	1C	AE	01	C1	000FA		ADDL3	#1, 28(SP), R0		
			57	50	C0	000FF		ADDL2	R0, WORK_PTR		
		30	AE	10	A9	9B	00102	MOVZBW	16(CHILD), VERB_NAME_DSC		
				32	AE	B4	00107	CLRW	VERB_NAME_DSC+2		
		34	AE	11	A9	9E	0010A	MOVAB	17(R9), VERB_NAME_DSC+4		
				5A	DD	0010F		PUSHL	R10		
				34	AE	9F	00111	PUSHAB	VERB_NAME_DSC		
		00000000G	00	02	FB	00114		CALLS	#2, CDU\$ADD VERB_NAME		
			59	04	A9	D0	0011B	MOVL	4(CHILD), CHILD		
				C7	11	0011F	8\$:	BRB	7\$		
			57	56	C2	00121	9\$:	SUBL2	VARIABLE_PTR, R7	0912	
			57	01	83	00124		SUBB3	#1, R7, (VARIABLE_PTR)		
	66			0A	11	00128		BRB	11\$	0878	
				01	C1	0012A	10\$:	ADDL3	#1, 28(SP), R0	0919	
	50	1C	AE	50	28	0012F		MOVC3	R0, 16(R11), (VARIABLE_PTR)		
	66	10	AB	5A	A3	00134	11\$:	SUBW3	R10, VARIABLE_PTR, 24(R10)	0924	
	AA		56	66	9A	00139		MOVZBL	(VARIABLE_PTR), R0	0925	
			50	01	A046	9E	0013C	MOVAB	1(R0)[VARIABLE_PTR], VARIABLE_PTR		
			56	08	AB	D0	00141	MOVL	8(R11), CHILD	1141	
			59	03	12	00145	12\$:	BNEQ	13\$		
				0203	31	00147		BRW	46\$		
				69	AF	0014A	13\$:	CASEW	(CHILD), #0, #53		
	35		00	006C		0014E	14\$:	.WORD	15\$-14\$,-		
006C	006C		006C	006C		00156			15\$-14\$,-		
007B	006C		006C	006C		0015E			15\$-14\$,-		
0107	00FB		00D7	00D1		00166			15\$-14\$,-		
0193	018D		010D	006C		0016E			15\$-14\$,-		
006C	01D5		01CE	019A		00176			15\$-14\$,-		
006C	006C		006C	006C		0017E			15\$-14\$,-		
006C	006C		006C	006C							

				06	12	001DD	BNEQ	19\$	
	14	BE		04	88	001DF	BISB2	#4, @20(SP)	
		24		34	11	001E3	BRB	23\$	
				67	B1	001E5	CMPW	(GRANDCHILD), #36	
	14	BE		06	12	001E8	BNEQ	20\$	
				08	88	001EA	BISB2	#8, @20(SP)	
		27		29	11	001EE	BRB	23\$	
				67	B1	001F0	CMPW	(GRANDCHILD), #39	
	14	BE		06	12	001F3	BNEQ	21\$	
				10	88	001F5	BISB2	#16, @20(SP)	
		28		1E	11	001F9	BRB	23\$	
				67	B1	001FB	CMPW	(GRANDCHILD), #40	
	14	BE		06	12	001FE	BNEQ	22\$	
				02	88	00200	BISB2	#2, @20(SP)	
		7E	02	13	11	00204	BRB	23\$	
				A7	3C	00206	MOVZWL	2(GRANDCHILD), -(SP)	
				01	DD	0020A	PUSHL	#1	
00000000G	00		00000000G	8F	DD	0020C	PUSHL	#CDU\$ IGNCLIFLAG	
	57			03	FB	00212	CALLS	#3, CDU\$REPORT_SEMANTIC_ERROR	
				A7	DD	00219	MOVL	4(GRANDCHILD), -GRANDCHILD	
	24	BE		AE	11	0021D	BRB	17\$	
				01	90	0021F	MOVB	#1, @36(SP)	
		57		28	11	00223	BRB	29\$	
				A9	DD	00225	MOVL	8(CHILD), GRANDCHILD	
				57	DD	00229	PUSHL	GRANDCHILD	
				5B	DD	0022B	PUSHL	R11	
00000000G	00			02	FB	0022D	CALLS	#2, CDU\$GENERATE_EXPRESSION	
0C	A9	0C		A7	DD	00234	MOVL	12(GRANDCHILD), T2(CHILD)	
14	BE	80		8F	88	00239	BISB2	#128, @20(SP)	
				58	D6	0023E	INCL	DISALLOW_COUNT	
				17	11	00240	BRB	31\$	
	14	BE	80	8F	88	00242	BISB2	#128, @20(SP)	
				10	11	00247	BRB	31\$	
	24	BE		03	90	00249	MOVB	#3, @36(SP)	
20	BE	56		5A	A3	0024D	SUBW3	R10, VARIABLE_PTR, @32(SP)	
				00DA	31	00252	BRW	44\$	
	04	AE		59	DD	00255	MOVL	CHILD, OUTPUTS_NODE	
				7E	11	00259	BRB	35\$	
50	10	BE		04	EF	0025B	EXTZV	#4, #4, @16(SP), R0	
				50	D6	00261	INCL	R0	
10	BE			50	F0	00263	INSV	R0, #4, #4, @16(SP)	
7E	10	BE		04	EF	00269	EXTZV	#4, #4, @16(SP), -(SP)	
				59	DD	0026F	PUSHL	CHILD	
				02	FB	00271	CALLS	#2, CDU\$GENERATE_ENTITY	
01	10	BE	00000000G	04	ED	00278	CMPZV	#4, #4, @16(SP), #1	
				0A	12	0027E	BNEQ	33\$	
		50	0C	A9	DD	00280	MOVL	12(CHILD), R0	
		08		50	DD	00284	MOVL	R0, 8(R10)	
				0C	11	00288	BRB	34\$	
		50	0C	A9	DD	0028A	MOVL	12(CHILD), R0	
	51	1C		08	C1	0028E	ADDL3	#8, LAST_PARM, R1	
				50	DD	00293	MOVL	R0, (R1)	
				00	DD	00296	ADDL2	CDU\$GL_TABLE, R0	
		1C	00000000G	50	DD	0029D	MOVL	R0, LAST_PARM	
		14		20	88	002A1	BISB2	#32, @20(SP)	
		04		04	E1	002A5	BBC	#4, 4(R0), 42\$	
50	10	77		00	EF	002AA	EXTZV	#0, #4, @16(SP), R0	
	BE	04							

10	BE					50	D6	002B0	INCL	R0		
	50	10	04			50	F0	002B2	INSV	R0, #0, #4, @16(SP)		
	50	10	BE			00	EF	002B8	EXTZV	#0, #4, @16(SP), R0		
						04	ED	002BE	CMPZV	#4, #4, @16(SP), R0		
						58	1B	002C4	BLEQU	42\$		
				7E	02	A9	3C	002C6	MOVZWL	2(CHILD), -(SP)		
						01	DD	002CA	PUSHL	#1		
					00000000G	8F	DD	002CC	PUSHL	#CDUS_INVREQPARM		
						03	FB	002D2	CALLS	#3, CDUS\$REPORT_SEMANTIC_ERROR		
						68	11	002D9	BRB	45\$		
				14	BE	20	88	002DB	BISB2	#32, @20(SP)		
						65	11	002DF	BRB	45\$		
						5A	A3	002E1	SUBW3	R10, VARIABLE_PTR, 30(R10)		
						47	11	002E6	BRB	44\$		
						08	AE	D6	INCL	QUAL_COUNTER		
						08	AE	DD	PUSHL	QUAL_COUNTER		
						59	DD	002EE	PUSHL	CHILD		
					00000000G	02	FB	002F0	CALLS	#2, CDUS\$GENERATE_ENTITY		
						01	AE	D1	CMPL	QUAL_COUNTER, #1		
						0A	12	002FB	BNEQ	39\$		
						50	D0	002FD	MOVL	12(CHILD), R0		
					OC	AA	D0	00301	MOVL	R0, 12(R10)		
						OC	11	00305	BRB	40\$		
						50	D0	00307	MOVL	12(CHILD), R0		
						08	C1	0030B	ADDL3	#8, LAST_QUAL, R1		
						50	D0	00310	MOVL	R0, (R1)		
						50	C1	00313	ADDL3	R0, CDUS\$GL_TABLE, LAST_QUAL		
						8F	88	0031C	BISB2	#64, @20(SP)		
						23	11	00321	BRB	45\$		
						02	90	00323	MOVB	#2, @36(SP)		
						5A	A3	00327	SUBW3	R10, VARIABLE_PTR, @32(SP)		
						04	C0	0032C	ADDL2	#4, VARIABLE_PTR		
						04	9A	0032F	MOVZBL	16(CHILD), 12(SP)		
						01	C1	00334	ADDL3	#1, 12(SP), R0		
						50	28	00339	MOVC3	R0, 16(CHILD), (VARIABLE_PTR)		
						01	C1	0033E	ADDL3	#1, 12(SP), R0		
						50	C0	00343	ADDL2	R0, VARIABLE_PTR		
						04	D0	00346	MOVL	4(CHILD), CHILD		
						FD	31	0034A	BRW	12\$		
						58	D5	0034D	TSTL	DISALLOW_COUNT		1147
						6D	13	0034F	BEQL	51\$		
						AE	9F	00351	PUSHAB	EXPRESSION		1148
						02	78	00354	ASHL	#2, DISALLOW_COUNT, 32(SP)		
						08	C0	00359	ADDL2	#8, 32(SP)		
						AE	9F	0035D	PUSHAB	32(SP)		
						02	FB	00360	CALLS	#2, LIB\$GET_VM		
						50	E8	00367	BLBS	STATUS, 47\$		
						50	DD	0036A	PUSHL	STATUS		
						01	FB	0036C	CALLS	#1, LIB\$SIGNAL		
						AE	D0	00373	MOVL	EXPRESSION, R0		1152
						8F	3C	00377	MOVZWL	#1285, 2(R0)		
						06	A0	0037D	CLRW	6(R0)		1155
						AB	D0	00380	MOVL	8(R11), CHILD		1169
						18	13	00384	BEQL	50\$		
						69	B1	00386	CMPW	(CHILD), #9		
						0D	12	00389	BNEQ	49\$		
						A0	3C	0038B	MOVZWL	6(R0), R1		

	08	A041	0C	A9	D0	0038F	MOVL	12(CHILD), 8(R0)[R1]			
			06	A0	B6	00395	INCL	6(R0)			
		59	04	A9	D0	00398	49\$:	MOVL	4(CHILD), CHILD		
				E6	11	0039C	BRB	48\$			
		58		04	C4	0039E	50\$:	MULL2	#4, R8	1174	
		58		0B	C0	003A1		ADDL2	#11, R8		
		58		04	C6	003A4		DIVL2	#4, R8		
60		58		04	A5	003A7		MULW3	#4, R8, (R0)		
		51	00000000G	00	D0	003AB		MOVL	CDU\$GL_TABLE, R1		
		52		60	3C	003B2		MOVZWL	(R0), R2		
10	AA	10		52	C0	003B5		ADDL2	R2, 16(R1)		
		50		51	C3	003B9		SUBL3	R1, R0, 16(R10)	1178	
				04	AE	D5	003BE	51\$:	TSTL	OUTPUTS_NODE	1183
				14	13	003C1		BEQL	52\$		
1C	AA	56		5A	A3	003C3		SUBW3	R10, VARIABLE_PTR, 28(R10)	1184	
				56	DD	003C8		PUSHL	VARIABLE_PTR	1186	
				08	AE	DD	003CA		PUSHL	OUTPUTS_NODE	
				5B	DD	003CD		PUSHL	R11		
	0000V	CF		03	FB	003CF		CALLS	#3, CDU\$GENERATE_OUTPUTS_LIST		
		56		50	C0	003D4		ADDL2	R0, VARIABLE_PTR		
				24	BE	95	003D7	52\$:	TSTB	@36(SP)	1195
				21	12	003DA		BNEQ	54\$		
		1A		6E	E9	003DC		BLBC	DOING_VERB, 53\$	1196	
			0000'	CF	9F	003DF		PUSHAB	P.AAC	1197	
	00000000G	00		01	FB	003E3		CALLS	#1, CL\$PRESENT		
		10		50	E8	003EA		BLBS	R0, 54\$		
		24		03	90	003ED		MOVB	#3, @36(SP)	1198	
20	BE	18		01	A1	003F1		ADDW3	#1, 24(R10), @32(SP)	1199	
				04	11	003F7		BRB	54\$	1196	
		24		04	90	003F9	53\$:	MOVB	#4, @36(SP)	1202	
				CF	9F	003FD	54\$:	PUSHAB	P.AAE	1204	
	00000000G	00	0000'	01	FB	00401		CALLS	#1, CL\$PRESENT		
		1A		50	E9	00408		BLBC	R0, 56\$		
		01	24	BE	91	0040B		CMPB	@36(SP), #1	1205	
				06	13	0040F		BEQL	55\$		
		03	24	BE	91	00411		CMPB	@36(SP), #3	1206	
				27	12	00415		BNEQ	58\$		
		7E	02	AB	3C	00417	55\$:	MOVZWL	2(R11), -(SP)	1207	
				01	DD	0041B		PUSHL	#1		
			00000000G	8F	DD	0041D		PUSHL	#CDU\$_ROUTREQ		
				12	11	00423		BRB	57\$		
		02	24	BE	91	00425	56\$:	CMPB	@36(SP), #2	1209	
				13	12	00429		BNEQ	58\$		
		7E	02	AB	3C	0042B		MOVZWL	2(R11), -(SP)	1210	
				01	DD	0042F		PUSHL	#1		
			00000000G	8F	DD	00431		PUSHL	#CDU\$_INVROUT		
	00000000G	00		03	FB	00437	57\$:	CALLS	#3, CDU\$REPORT_SEMANTIC_ERROR		
		56		5A	C2	0043E	58\$:	SUBL2	R10, R6	1214	
		56		03	C0	00441		ADDL2	#3, R6		
		56		04	C6	00444		DIVL2	#4, R6		
6A		56		04	A5	00447		MULW3	#4, R6, (R10)		
		50	00000000G	00	D0	0044B		MOVL	CDU\$GL_TABLE, R0		
		51		6A	3C	00452		MOVZWL	(R10), R1		
				51	C0	00455		ADDL2	R1, 16(R0)		
0C	AB	10		50	C3	00459		SUBL3	R0, R10, 12(R11)	1218	
				04	0045E		RET			1221	

GENCODE2
V04-000

D 11
15-Sep-1984 23:37:28
14-Sep-1984 11:58:21

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[CDU.SRC]GENCODE2.B32;1 Page 17
(6)

; Routine Size: 1119 bytes, Routine Base: \$CODE\$ + 0000

```

: 491      1222 1  !++
: 492      1223 1  Description: This routine is called when an outputs list must be added
: 493      1224 1  to a command block. An outputs list is the result of an
: 494      1225 1  OUTPUTS clause in the CLD, which contains a list of
: 495      1226 1  parameters or qualifiers which specify output files.
: 496      1227 1  Such clauses are only needed for the old CLI interface.
: 497      1228 1
: 498      1229 1  The outputs list consists of a counted sequence of bytes,
: 499      1230 1  one for each item in the OUTPUTS clause. Each byte
: 500      1231 1  contains the negative of the parameter number (for
: 501      1232 1  parameters), or the qualifier number (for qualifiers).
: 502      1233 1
: 503      1234 1  Parameters: parent      By reference, the parent of the outputs
: 504      1235 1  outputs      node, which has the parameters and qualifiers
: 505      1236 1  outputs_list By reference, the outputs node.
: 506      1237 1  outputs_list By reference, the location which is to
: 507      1238 1  receive the outputs list.
: 508      1239 1
: 509      1240 1
: 510      1241 1  Returns:      By value, the length of the outputs list.
: 511      1242 1
: 512      1243 1  Notes:
: 513      1244 1  --
: 514      1245 1
: 515      1246 1  GLOBAL ROUTINE cdu$generate_outputs_list(parent: ref node,
: 516      1247 1  outputs: ref node,
: 517      1248 1  outputs_list: pointer)
: 518      1249 2  = BEGIN
: 519      1250 2
: 520      1251 2  local
: 521      1252 2  outputs_ptr: pointer,
: 522      1253 2  outputs_item: ref node,
: 523      1254 2  entity: ref node;
: 524      1255 2
: 525      1256 2
: 526      1257 2  ! Scan each of the children of the outputs node, placing one byte in the
: 527      1258 2  ! outputs list for each one.
: 528      1259 2
: 529      1260 2  outputs_ptr = .outputs_list + 1;
: 530      1261 2  scan_children(outputs,outputs_item,
: 531      1262 2
: 532      1263 2  ! Each output item specifies an entity which is an output of the
: 533      1264 2  ! command. The entity is specified by its label (as given in a
: 534      1265 2  ! LABEL clause), or, if there is no label, by its name. Scan the
: 535      1266 2  ! children of the parent node looking for the one which represents
: 536      1267 2  ! the output entity.
: 537      1268 2
: 538      1269 2  scan_children(parent,entity,
: 539      1270 2
: 540      1271 2  if .entity[node_w_type] eqlu node_k_parameter or
: 541      1272 2  .entity[node_w_type] eqlu node_k_qualifier then (
: 542      1273 2
: 543      1274 2  if cdu$lookup_child(.entity,node_k_label,.outputs_item[node_b_text_length],
: 544      1275 2  outputs_item[node_t_text]) neqa 0 then
: 545      1276 2  exitloop
: 546      1277 2  else if ch$eq(.outputs_item[node_b_text_length],outputs_item[node_t_text],
: 547      1278 2  .entity[node_b_text_length],entity[node_t_text],%x'00') then
```



```
548 P 1279 2 exitloop;
549 P 1280 2 );
550 P 1281 2 );
551 P 1282 2 ! If we found the parameter or qualifier, then set up a pointer to
552 P 1283 2 ! the entity block generated for it.
553 P 1284 2
554 P 1285 2 if .entity neq 0 then (
555 P 1286 2 bind
556 P 1287 2 entity_block = .cdu$gl_table + .entity[node_l_code]: block[,byte];
557 P 1288 2
558 P 1289 2 ! If the entity is a parameter, store the negative of its
559 P 1290 2 ! number in the outputs list. If a qualifier, store the
560 P 1291 2 ! qualifier number.
561 P 1292 2
562 P 1293 2 outputs_ptr[0,0,8,1] = (if .entity_block[ent_b_subtype] eqv ent_k_parameter then
563 P 1294 2 -.entity_block[ent_b_number]
564 P 1295 2 else
565 P 1296 2 .entity_block[ent_b_number]);
566 P 1297 2
567 P 1298 2 increment(outputs_ptr);
568 P 1299 2
569 P 1300 2 ) else
570 P 1301 2
571 P 1302 2 ! We didn't find the parameter or qualifier.
572 P 1303 2
573 P 1304 2 cdu$report_semantic_error(msg(cdu$_undefoutput),2,.outputs_item[node_w_line],
574 P 1305 2 outputs_item[node_b_text_length]);
575 P 1306 2 );
576 P 1307 2
577 P 1308 2 ! Store the count of outputs at the front of the outputs list.
578 P 1309 2
579 P 1310 2 outputs_list[0,0,8,0] = .outputs_ptr - .outputs_list - 1;
580 P 1311 2
581 P 1312 2 ! Return the length of the outputs list.
582 P 1313 2
583 P 1314 2 return 1+.outputs_list[0,0,8,0];
584 P 1315 2
585 P 1316 1 END;
```

					00FC 0000	.EXTRN	CDU\$_UNDEFOUTPUT		
						.ENTRY	CDU\$GENERATE_OUTPUTS_LIST, Save R2,R3,R4,-	1246	
							R5,R6,R7		
54	0C	AC	01	C1	00002	ADDL3	#1, OUTPUTS_LIST, OUTPUTS_PTR	1260	
		50	08	AC	D0 00007	MOVL	OUTPUTS, R0	1306	
		55	08	A0	D0 0000B	MOVL	8(R0), OUTPUTS_ITEM		
		57	04	AC	D0 0000F	MOVL	PARENT, R7		
				55	D5 00013	TSTL	OUTPUTS_ITEM		
				03	12 00015	BNEQ	2\$		
				0080	31 00017	BRW	11\$		
		56	08	A7	D0 0001A	MOVL	8(R7), ENTITY		
				38	13 0001E	BEQL	6\$		
		0D		66	B1 00020	CMPW	(ENTITY), #13		
				05	13 00023	BEQL	4\$		
		10		66	B1 00025	CMPW	(ENTITY), #16		

50	00	11	51	10	28	12	00028	BNEQ	5\$	:
		7E	10	10	A5	9F	0002A	4\$: PUSHAB	17(OUTPUTS_ITEM)	:
					A5	9A	0002D	MOVZBL	16(OUTPUTS_ITEM), -(SP)	:
					1A	DD	00031	PUSHL	#26	:
					56	DD	00033	PUSHL	ENTITY	:
	00000000G	00			04	FB	00035	CALLS	#4, CDU\$LOOKUP_CHILD	:
					50	D5	0003C	TSTL	R0	:
					18	12	0003E	BNEQ	6\$	:
		51	10	10	A5	9A	00040	MOVZBL	16(OUTPUTS_ITEM), R1	:
		50	10	10	A6	9A	00044	MOVZBL	16(ENTITY), R0	:
		A5		11	51	2D	00048	CMPC5	R1, 17(OUTPUTS_ITEM), #0, R0, 17(ENTITY)	:
					A6		0004E			:
		56	04	04	06	13	00050	BEQL	6\$	:
					A6	D0	00052	5\$: MOVL	4(ENTITY), ENTITY	:
					C6	11	00056	BRB	3\$	:
					56	D5	00058	6\$: TSTL	ENTITY	:
					21	13	0005A	BEQL	9\$	:
50	00000000G	00	0C	0C	A6	C1	0005C	ADDL3	12(ENTITY), CDU\$GL_TABLE, R0	:
		01	03	03	A0	91	00065	CMPB	3(R0), #1	:
					09	12	00069	BNEQ	7\$	:
		50	14	14	A0	9A	0006B	MOVZBL	20(R0), R0	:
		50			50	CE	0006F	MNEGL	R0, R0	:
					04	11	00072	BRB	8\$	:
		50	14	14	A0	9A	00074	7\$: MOVZBL	20(R0), R0	:
		84			50	90	00078	8\$: MOVB	R0, (OUTPUTS_PTR)+	:
					16	11	0007B	BRB	10\$	:
			10	10	A5	9F	0007D	9\$: PUSHAB	16(OUTPUTS_ITEM)	:
		7E	02	02	A5	3C	00080	MOVZWL	2(OUTPUTS_ITEM), -(SP)	:
					02	DD	00084	PUSHL	#2	:
					8F	DD	00086	PUSHL	#CDU\$ UNDEFOUTPUT	:
	00000000G	00			04	FB	0008C	CALLS	#4, CDU\$REPORT_SEMANTIC_ERROR	:
		55	04	04	A5	D0	00093	10\$: MOVL	4(OUTPUTS_ITEM), OUTPUTS_ITEM	:
					FF79	31	00097	BRW	1\$	:
		54	0C	0C	AC	C2	0009A	11\$: SUBL2	OUTPUTS_LIST, R4	: 1310
0C	BC	54			01	83	0009E	SUBB3	#1, R4, @OUTPUTS_LIST	: 1314
		50	0C	0C	BC	9A	000A3	MOVZBL	@OUTPUTS_LIST, R0	: 1316
					50	D6	000A7	INCL	R0	:
					04	000A9		RET		:

; Routine Size: 170 bytes, Routine Base: \$CODE\$ + 045F


```
587 1317 1 !++
588 1318 1 ! Description: This routine is called to generate a type block and the list
589 1319 1 ! of entity blocks which represent the keywords. The entity
590 1320 1 ! blocks are linked together and hung off the type block.
591 1321 1
592 1322 1 ! Parameters: top_node By reference, the node that represents the
593 1323 1 ! type definition. Its children are the
594 1324 1 ! keyword nodes.
595 1325 1
596 1326 1 ! Returns: Nothing.
597 1327 1
598 1328 1 ! Notes:
599 1329 1 ! --
600 1330 1
601 1331 1 GLOBAL ROUTINE cdu$generate_type(top_node: ref node) : novalue
602 1332 2 = BEGIN
603 1333 2
604 1334 2 local
605 1335 2     status: long,
606 1336 2     type: pointer,
607 1337 2     variable_ptr: pointer,
608 1338 2     child: ref node,
609 1339 2     keyword_counter: long initial(0),
610 1340 2     last_block: pointer;
611 1341 2
612 1342 2
613 1343 2 ! Allocate enough space to contain the largest possible type block.
614 1344 2
615 1345 2 allocate_largest_table_block(type_k_length + type_k_max_name + type_k_max_prefix, type);
616 1346 2
617 1347 2 ! Begin by initializing the type block. This includes any fields that
618 1348 2 ! don't depend on the intermediate representation.
619 1349 2
620 1350 2 type[type_b_type] = block_k_type;
621 1351 2 type[type_b_subtype] = type_k_type;
622 1352 2 type[type_w_flags] = 0;
623 1353 2 type[type_w_tro_count] = 1;
624 1354 2 type[type_w_name] = type[type_w_prefix] = 0;
625 1355 2
626 1356 2 ! Set up to add information to the variable portion of the block.
627 1357 2
628 1358 2 variable_ptr = type[type_z_variable];
629 1359 2
630 1360 2 ! Add the type name to the type block as an ASCII string.
631 1361 2
632 1362 2 type[type_w_name] = .variable_ptr - .type;
633 1363 2 ch$move(1+.top_node[node_b_text_length],top_node[node_b_text_length],
634 1364 2     .variable_ptr);
635 1365 2 variable_ptr = .variable_ptr + 1+.top_node[node_b_text_length];
636 1366 2
637 1367 2 ! Scan the children of the node representing the type definition.
638 1368 2
639 1369 2 scan_children(top_node,child,
P 1370 2
640 1371 2     ! Determine our action based on the type of child node.
P 1372 2
641 1373 2     selectoneu .child[node_w_type] of set
P 1373 2
```

```

; INFO#250                               LI:1409
; Referenced LOCAL symbol LAST_BLOCK is probably not initialized

```

```

OFFC 00000      .ENTRY CDUS$GENERATE TYPE, Save R2,R3,R4,R5,R6,R7,- : 1331
                  R8,R9,R10,R11                                     :
5E              OC  C2 00002      SUBL2 #12, $P                                     :

```


			04	AE	D4	00005	CLRL	KEYWORD_COUNTER	:	1332	
			08	AE	9F	00008	PUSHAB	TYPE	:	1345	
		04	50	8F	9A	0000B	MOVZBL	#80, 4(SP)	:		
			04	AE	9F	00010	PUSHAB	4(SP)	:		
		00000000G	00	02	FB	00013	CALLS	#2, LIB\$GET_VM	:		
			09	50	E8	0001A	BLBS	STATUS, 1\$	:		
				50	DD	0001D	PUSHL	STATUS	:		
		00000000G	00	01	FB	0001F	CALLS	#1, LIB\$SIGNAL	:		
			58	08	AE	D0	00026	1\$:	MOV	TYPE, R8	1350
		02	A8	0103	8F	3C	0002A	MOVZWL	#259, 2(R8)	:	
		06	A8		01	B0	00030	MOVW	#1, 6(R8)	:	1353
				0C	A8	D4	00034	CLRL	12(R8)	:	1354
			56	10	A8	9E	00037	MOVAB	16(R8), VARIABLE_PTR	:	1358
OC	A8		56		58	A3	0003B	SUBW3	R8, VARIABLE_PTR, 12(R8)	:	1362
			5A	04	AC	D0	00040	MOV	TOP_NODE, R10	:	1363
			57	10	AA	9A	00044	MOVZBL	16(R10), R7	:	
			50	01	A7	9E	00048	MOVAB	1(R7), R0	:	
	66	10	AA		50	28	0004C	MOV3	R0, 16(R10), (VARIABLE_PTR)	:	1364
			56	01	A7	9E	00051	MOVAB	1(R7)[VARIABLE_PTR], VARIABLE_PTR	:	1365
			57	08	AA	D0	00056	MOV	8(R10), CHILD	:	1409
					67	13	0005A	2\$:	BEQ	8\$	
			0F		67	B1	0005C	CMPW	(CHILD), #15	:	
					19	12	0005F	BNEQ	3\$	:	
OE	A8		56		58	A3	00061	SUBW3	R8, VARIABLE_PTR, 14(R8)	:	
			59	10	A7	9A	00066	MOVZBL	16(CHILD), R9	:	
			50	01	A9	9E	0006A	MOVAB	1(R9), R0	:	
	66	10	A7		50	28	0006E	MOV3	R0, 16(CHILD), (VARIABLE_PTR)	:	
			56	01	A9	9E	00073	MOVAB	1(R9)[VARIABLE_PTR], VARIABLE_PTR	:	
					43	11	00078	BRB	7\$	:	
			18		67	B1	0007A	3\$:	CMPW	(CHILD), #24	
					31	12	0007D	BNEQ	6\$	:	
				04	AE	D6	0007F	INCL	KEYWORD_COUNTER	:	
				04	AE	DD	00082	PUSHL	KEYWORD_COUNTER	:	
					57	DD	00085	PUSHL	CHILD	:	
		00000000G	00		02	FB	00087	CALLS	#2, CDUS\$GENERATE_ENTITY	:	
			01	04	AE	D1	0008E	CMPL	KEYWORD_COUNTER, #1	:	
					0A	12	00092	BNEQ	4\$	:	
			50	0C	A7	D0	00094	MOV	12(CHILD), R0	:	
		08	A8		50	D0	00098	MOV	R0, 8(R8)	:	
					08	11	0009C	BRB	5\$	:	
			50	0C	A7	D0	0009E	4\$:	MOV	12(CHILD), R0	
		08	AB		50	D0	000A2	5\$:	MOV	R0, 8(LAST_BLOCK)	
5B	00000000G	00			50	C1	000A6	6\$:	ADDL3	R0, CDUS\$GL_TABLE, LAST_BLOCK	
					0D	11	000AE	7\$:	BRB	7\$	
		00000000G	00		8F	DD	000B0	8\$:	PUSHL	#CDUS\$ INTINVNODE	
			57	04	01	FB	000B6	CALLS	#1, LIB\$SIGNAL	:	
					A7	D0	000BD	MOV	4(CHILD), CHILD	:	
			56		97	11	000C1	BRB	2\$	:	
			56		58	C2	000C3	SUBL2	R8, R6	:	1413
			56		03	C0	000C6	ADDL2	#3, R6	:	
			56		04	C6	000C9	DIVL2	#4, R6	:	
68			56		04	A5	000CC	MULW3	#4, R6, (R8)	:	
			50	00000000G	00	D0	000D0	MOV	CDUS\$GL_TABLE, R0	:	
			51		68	3C	000D7	MOVZWL	(R8), R1	:	
		10	A0		51	C0	000DA	ADDL2	R1, 16(R0)	:	1417
OC	AA		58		50	C3	000DE	SUBL3	R0, R8, 12(R10)	:	1420
					04	000E3	RET			:	

; Routine Size: 228 bytes, Routine Base: \$CODE\$ + 0509

```

: 691      1421  1
: 692      1422  1 END
: 693      1423  0 ELUDOM

```

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$GLOBAL\$	1	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$SPLITS	48	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODE\$	1517	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	4	0	1000	00:01.8

```

: Information: 3
: Warnings: 0
: Errors: 0

```

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:GENCODE2/OBJ=OBJ\$:GENCODE2 MSRC\$:GENCODE2/UPDATE=(ENH\$:GENCODE2)

```

: Size:      1517 code + 49 data bytes
: Run Time:   00:35.9
: Elapsed Time: 01:12.9
: Lines/CPU Min: 2377
: Lexemes/CPU-Min: 24467
: Memory Used: 399 pages
: Compilation Complete

```


0043 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

GENRAL REQ R32	EXTCAL LIS
CLISDEF R32	GENCODE4 LIS
CDUMSGS LIS	GENCODE1 LIS
CDU	GENCODE3 LIS
CDU MAP	GENCODE2 LIS
CDUREQ R32	CDUTYPEDEF LIS