

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
CCCCCCCCCCCC	DDDDDDDDDDDD	UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU

```

PPPPPPPP      AAAAAA      RRRRRRRR      SSSSSSSS      EEEEEEEEEEE      222222
PPPPPPPP      AAAAAA      RRRRRRRR      SSSSSSSS      EEEEEEEEEEE      222222
PP           PP  AA          AA  RR          RR  SS           EE           22           22
PP           PP  AA          AA  RR          RR  SS           EE           22           22
PP           PP  AA          AA  RR          RR  SS           EE           22           22
PP           PP  AA          AA  RR          RR  SS           EE           22           22
PPPPPPPP      AA          AA  RRRRRRRR      SSSSSS      EEEEEEEE      22
PPPPPPPP      AA          AA  RRRRRRRR      SSSSSS      EEEEEEEE      22
PP           AAAAAAAAAA  RR  RR          SS           EE           22
PP           AAAAAAAAAA  RR  RR          SS           EE           22
PP           AA          AA  RR          RR          SS           EE           22
PP           AA          AA  RR          RR          SS           EE           22
PP           AA          AA  RR          RR          SS           EE           22
PP           AA          AA  RR          RR          SSSSSSSS      EEEEEEEEEEE      2222222222
PP           AA          AA  RR          RR          SSSSSSSS      EEEEEEEEEEE      2222222222

```

[illegible]

```

1 0001 0 MODULE parse2 (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, CLD Parser Module 2
30 0030 1
31 0031 1 Abstract: This module is one of a few modules that implements the
32 0032 1 parser for CLD files. This parser translates the CLD source
33 0033 1 language into an intermediate representation composed of
34 0034 1 nodes linked in a directed graph.
35 0035 1
36 0036 1 Environment: Standard CDU environment.
37 0037 1
38 0038 1 Author: Paul C. Anagnostopoulos
39 0039 1 Creation: 6 December 1982
40 0040 1
41 0041 1 Modifications:
42 0042 1 --
43 0043 1
44 0044 1
45 0045 1 Library 'sys$library:lib';
46 0046 1 require 'clitabdef';
47 0371 1 require 'cdureq';

```

49	0785	1	!	T A B L E O F C O N T E N T S
50	0786	1	!	-----
51	0787	1		
52	0788	1	forward	routine
53	0789	1		cdu\$param_clause,
54	0790	1		cdu\$qual_clause,
55	0791	1		cdu\$type_clause,
56	0792	1		cdu\$keyword_clause,
57	0793	1		cdu\$common_clause,
58	0794	1		cdu\$value_clause,
59	0795	1		cdu\$builtin_type,
60	0796	1		cdu\$cli_flag;
61	0797	1		
62	0798	1		
63	0799	1	!	E X T E R N A L R E F E R E N C E S
64	0800	1	!	-----
65	0801	1		
66	0802	1	external	routine
67	0803	1		cdu\$check_for_children,
68	0804	1		cdu\$create_node,
69	0805	1		cdu\$get_next_token,
70	0806	1		cdu\$lookup_child,
71	0807	1		cdu\$report_syntax_error,
72	0808	1		cdu\$token_must_be;
73	0809	1		
74	0810	1	external	
75	0811	1		cdu\$gl_token_class: long,
76	0812	1		cdu\$gq_token: descriptor;

```

78 0813 1  !++
79 0814 1  ! Description: This syntax routine recognizes the "param-clause" construct,
80 0815 1  ! which is used to define the characteristics of a parameter.
81 0816 1  !
82 0817 1  ! Parameters: parent By reference, parent node of this construct.
83 0818 1  !
84 0819 1  ! Returns: The top-level node representing the parameter clause,
85 0820 1  ! or zero if there aren't any more.
86 0821 1  !
87 0822 1  !
88 0823 1  ! Notes:
89 0824 1  ! --
90 0825 1  !
91 0826 1  GLOBAL ROUTINE cdu$param_clause(parent: ref node)
92 0827 2  = BEGIN
93 0828 2  local
94 0829 2  clause: ref node;
95 0830 2
96 0831 2
97 0832 2
98 0833 2  ! Determine which kind of clause we have.
99 0834 2
100 0835 2  if token_is(tkn_k_symbol,'PROMPT') then (
101 0836 2
102 0837 2      ! We have an PROMPT clause. It conflicts with any existing clause.
103 0838 2
104 0839 2      if cdu$check_for_children(.parent,node_k_prompt) then
105 0840 2          cdu$report_syntax_error(msg(cdu$_dupprompt));
106 0841 2
107 0842 2      ! Bypass any optional equal sign.
108 0843 2
109 0844 2      cdu$get_next_token();
110 0845 2      skip_optional_token(tkn_k_equal);
111 0846 2
112 0847 2      ! Now we have a symbol or string which is the prompt. Store the
113 0848 2      ! prompt in a node.
114 0849 2
115 0850 2      clause = cdu$create_node(node_k_prompt,.cdu$gq_token[len],.cdu$gq_token[ptr]);
116 0851 2      if token_is(tkn_k_symbol) then
117 0852 2          cdu$get_next_token()
118 0853 2      else
119 0854 2          cdu$token_must_be(tkn_k_string);
120 0855 2      return .clause;
121 0856 2  );
122 0857 2
123 0858 2  ! We either have a common clause now, or we've run out of clauses.
124 0859 2
125 0860 2  return cdu$common_clause(.parent);
126 0861 2
127 0862 1  END;

```

```

.TITLE PARSE2
.IDENT \V04-000\
.PSECT $SPLITS,NOWRT,NOEXE,2

```

54 50 4D 4F 52 50 00000 P.AAA: .ASCII \PROMPT\

.EXTRN CDUSCHECK_FOR_CHILDREN
.EXTRN CDUSCREATE_NODE
.EXTRN CDUSGET_NEXT_TOKEN
.EXTRN CDUSLOOKUP_CHILD
.EXTRN CDUSREPORT_SYNTAX_ERROR
.EXTRN CDUSTOKEN_MUST_BE
.EXTRN CDUSGL_TOKEN_CLASS
.EXTRN CDUSGQ_TOKEN, CDUS_DUPPROMPT

.PSECT \$CODE\$,NOWRT,2

06	00	56	00000000G	00	9E	00002	.ENTRY	CDUSPARAM CLAUSE, Save R2,R3,R4,R5,R6	: 0826
		55	00000000G	00	9E	00009	MOVAB	CDUSGL_TOKEN_CLASS, R6	:
		54	00000000G	00	9E	00010	MOVAB	CDUSGET_NEXT_TOKEN, R5	:
		0D		66	D1	00017	MOVAB	CDUSGQ_TOKEN+4, R4	:
				5E	12	0001A	CMPL	CDUSGL_TOKEN_CLASS, #13	: 0835
		50		64	D0	0001C	BNEQ	5\$	:
		60	FC	A4	2D	0001F	MOVL	CDUSGQ_TOKEN+4, R0	:
			0000'	CF		00025	CMPCS	CDUSGQ_TOKEN, (R0), #0, #6, P.AAA	:
				50	12	00028	BNEQ	5\$	:
				13	DD	0002A	PUSHL	#19	: 0839
			04	AC	DD	0002C	PUSHL	PARENT	:
		00000000G	00	02	FB	00C2F	CALLS	#2, CDUSCHECK_FOR_CHILDREN	:
			0D	50	E9	00036	BLBC	R0, 1\$	:
		00000000G	00	8F	DD	00039	PUSHL	#CDUS_DUPPROMPT	: 0840
			65	01	FB	0003F	CALLS	#1, CDUSREPORT_SYNTAX_ERROR	:
			06	00	FB	00046	CALLS	#0, CDUSGET_NEXT_TOKEN	: 0844
				66	D1	00049	CMPL	CDUSGL_TOKEN_CLASS, #6	: 0845
				03	12	0004C	BNEQ	2\$	:
			65	00	FB	0004E	CALLS	#0, CDUSGET_NEXT_TOKEN	:
				64	DD	00051	PUSHL	CDUSGQ_TOKEN+4	: 0850
			7E	A4	3C	00053	MOVZWL	CDUSGQ_TOKEN, -(SP)	:
				13	DD	00057	PUSHL	#19	:
		00000000G	00	03	FB	00059	CALLS	#3, CDUSCREATE_NODE	:
			52	50	D0	00060	MOVL	R0, CLAUSE	:
			0D	66	D1	00063	CMPL	CDUSGL_TOKEN_CLASS, #13	: 0851
				05	12	00066	BNEQ	3\$	:
			65	00	FB	00068	CALLS	#0, CDUSGET_NEXT_TOKEN	: 0852
				09	11	0006B	BRB	4\$	:
				0B	DD	0006D	PUSHL	#11	: 0854
		00000000G	00	01	FB	0006F	CALLS	#1, CDUSTOKEN_MUST_BE	:
			50	52	D0	00076	MOVL	CLAUSE, R0	: 0855
				04		00079	RET		:
				AC	DD	0007A	PUSHL	PARENT	: 0860
		0000V	CF	01	FB	0007D	CALLS	#1, CDUSCOMMON_CLAUSE	:
				04		00G82	RET		: 0862

; Routine Size: 131 bytes, Routine Base: \$CODE\$ + 0000

```

129 0863 1  !++
130 0864 1  Description: This syntax routine parses the "qual-clause" construct,
131 0865 1  which is used to describe the characteristics of a qualifier.
132 0866 1
133 0867 1  Parameters: parent By reference, parent node of this construct.
134 0868 1
135 0869 1  Returns: The top-level node representing the qualifier clause,
136 0870 1  or zero if there aren't any more.
137 0871 1
138 0872 1  Notes:
139 0873 1  --
140 0874 1
141 0875 1 GLOBAL ROUTINE cdu$qual_clause(parent: ref node)
142 0876 2 = BEGIN
143 0877 2
144 0878 2 local
145 0879 2     clause: ref node;
146 0880 2
147 0881 2
148 0882 2 ! Determine which kind of clause we have.
149 0883 2
150 0884 2 if token_is(tkn_k_symbol,'BATCH') then (
151 0885 2
152 0886 2     ! We have a BATCH clause. It is represented by a node.
153 0887 2
154 0888 2     cdu$get_next_token();
155 0889 2     return cdu$create_node(node_k_batch);
156 0890 2 );
157 0891 2
158 0892 2 if token_is(tkn_k_symbol,'NEGATABLE') then (
159 0893 2
160 0894 2     ! We have a NEGATABLE clause. It is represented by a node.
161 0895 2     ! This clause conflicts with any existing NONNEGATABLE clause.
162 0896 2
163 0897 2     if cdu$check_for_children(.parent,node_k_nonnegatable) then
164 0898 2         cdu$report_syntax_error(msg(cdu$_confnonneg));
165 0899 2
166 0900 2     cdu$get_next_token();
167 0901 2     return cdu$create_node(node_k_negatable);
168 0902 2 );
169 0903 2
170 0904 2 if token_is(tkn_k_symbol,'NONNEGATABLE') then (
171 0905 2
172 0906 2     ! We have a NONNEGATABLE clause. It is represented by a node.
173 0907 2     ! This clause conflicts with any existing NEGATABLE clause.
174 0908 2
175 0909 2     if cdu$check_for_children(.parent,node_k_negatable) then
176 0910 2         cdu$report_syntax_error(msg(cdu$_confneg));
177 0911 2
178 0912 2     cdu$get_next_token();
179 0913 2     return cdu$create_node(node_k_nonnegatable);
180 0914 2 );
181 0915 2
182 0916 2 if token_is(tkn_k_symbol,'PLACEMENT') then (
183 0917 2
184 0918 2     ! We have an PLACEMENT clause. It conflicts with any existing clause.
185 0919 2

```

```

186 0920 if cdu$check_for_children(.parent,node_k_placement) then
187 0921     cdu$report_syntax_error(msg(cdu$_duplace));
188 0922
189 0923 ! Bypass any optional equal sign.
190 0924
191 0925 cdu$get_next_token();
192 0926 skip_optional_token(tkn_k_equal);
193 0927
194 0928 ! Now we have a symbol which tell us where the qualifier can appear
195 0929 ! on the command line. Create a node with the symbol
196 0930
197 0931 if not token_is(tkn_k_symbol,'GLOBAL') and
198 0932     not token_is(tkn_k_symbol,'LOCAL') and
199 0933     not token_is(tkn_k_symbol,'POSITIONAL') then
200 0934     cdu$report_syntax_error(msg(cdu$_invplace));
201 0935 clause = cdu$create_node(node_k_placement,.cdu$gq_token[ptr]);
202 0936 cdu$token_must_be(tkn_k_symbol);
203 0937 return .clause;
204 0938 );
205 0939
206 0940 ! We either have a common clause now, or we've run out of clauses.
207 0941
208 0942 return cdu$common_clause(.parent);
209 0943
210 0944 1 END;

```

```

.PSECT $SPLITS,NOWRT,NOEXE,2
45 4C 42 45 4C 42 41 54 41 47 45 4E 00006 P.AAB: .ASCII \BATCH\
54 4E 45 4D 45 43 41 4C 50 0000B P.AAC: .ASCII \NEGATABLE\
54 4E 45 4D 45 43 41 4C 50 00014 P.AAD: .ASCII \NONNEGATABLE\
4C 41 4E 4F 49 54 49 53 4F 50 00020 P.AAE: .ASCII \PLACEMENT\
4C 41 4E 4F 49 54 49 53 4F 50 00029 P.AAF: .ASCII \GLOBAL\
4C 41 4E 4F 49 54 49 53 4F 50 0002F P.AAG: .ASCII \LOCAL\
4C 41 4E 4F 49 54 49 53 4F 50 00034 P.AAH: .ASCII \POSITIONAL\

.EXTRN CDU$_CONFNONNEG
.EXTRN CDU$_CONFNEG, CDU$_DUPPLACE
.EXTRN CDU$_INVPLACE

.PSECT $CODE$,NOWRT,2
OFFC 00000
5B 00000000G 00 9E 00002 MOVAB CDU$QUAL CLAUSE, Save R2,R3,R4,R5,R6,R7,R8,-; 0875
5A 00000000G 00 9E 00009 MOVAB CDU$CREATE_NODE, R11
59 00000000G 00 9E 00010 MOVAB CDU$CHECK_FOR_CHILDREN, R10
58 00000000G 00 9E 00015 MOVAB P.AAB, R9
57 00000000G 00 9E 0001C MOVAB CDU$REPORT_SYNTAX_ERROR, R8
56 00000000G 00 9E 00023 MOVAB CDU$GET_NEXT_TOKEN, R7
55 00000000G 00 9E 0002A MOVAB CDU$GL_TOKEN_CLASS, R6
0D 00000000G 66 D1 00031 MOVAB CDU$GQ_TOKEN+4, R5
13 12 00034 CMPL CDU$GL_TOKEN_CLASS, #13
50 65 D0 00036 BNEQ 1$
05 00 60 FC A5 2D 00039 MOVL CDU$GQ_TOKEN+4, R0
CMPC5 CDU$GQ_TOKEN, (R0), #0, #5, P.AAB

```


PARSE2
V04-000

G 10
15-Sep-1984 23:48:10
14-Sep-1984 11:58:26

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[CDU.SRC]PARSE2.B32;1

Page 8
(4)

06	00	60	FC	A5	2D	000E2	CMPC5	CDUSGQ_TOKEN, (R0), #0, #6, P.AAF	:
			23	A9		000E8			:
		1D		29	13	000EA	BEQL	12\$	:
		50		54	E9	000EC	BLBC	R4, 11\$	0932
05	00	60	FC	65	D0	000EF	MOVL	CDUSGQ_TOKEN+4, R0	:
			29	A5	2D	000F2	CMPC5	CDUSGQ_TOKEN, (R0), #0, #5, P.AAG	:
				A9		000F8			:
		0D		19	13	000FA	BEQL	12\$	:
		50		54	E9	000FC	BLBC	R4, 11\$	0933
0A	00	60	FC	65	D0	000FF	MOVL	CDUSGQ_TOKEN+4, R0	:
			2E	A5	2D	00102	CMPC5	CDUSGQ_TOKEN, (R0), #0, #10, P.AAH	:
				A9		00108			:
				09	13	0010A	BEQL	12\$	:
		00000000G		8F	DD	0010C	PUSHL	#CDUS_INVPLACE	0934
		68		01	FB	00112	CALLS	#1, CDUSREPORT_SYNTAX_ERROR	:
				65	DD	00115	PUSHL	CDUSGQ_TOKEN+4	0935
		7E	FC	A5	3C	00117	MOVZWL	CDUSGQ_TOKEN, -(SP)	:
				17	DD	0011B	PUSHL	#23	:
		68		03	FB	0011D	CALLS	#3, CDUSCREATE_NODE	:
		52		50	D0	00120	MOVL	R0, CLAUSE	:
				0D	DD	00123	PUSHL	#13	0936
	00000000G	00		01	FB	00125	CALLS	#1, CDUSTOKEN_MUST_BE	:
		50		52	D0	0012C	MOVL	CLAUSE, R0	0937
				04	0012F		RET		:
			04	AC	DD	00130	PUSHL	PARENT	0942
	0000V	CF		01	FB	00133	CALLS	#1, CDUSCOMMON_CLAUSE	:
				04	00138		RET		0944

; Routine Size: 313 bytes, Routine Base: \$CODE\$ + 0083

```

212 0945 1  !++
213 0946 1  ! Description: This syntax routine recognizes the "type-clause" construct,
214 0947 1  ! used to define the characteristics of a data type.
215 0948 1  !
216 0949 1  ! Parameters: parent          By reference, parent node of this construct.
217 0950 1  !
218 0951 1  ! Returns:      The top-level node representing the type clause, or zero
219 0952 1  ! if there aren't any more.
220 0953 1  !
221 0954 1  ! Notes:
222 0955 1  ! --
223 0956 1  !
224 0957 1  GLOBAL ROUTINE cdu$type_clause(parent: ref node)
225 0958 2  = BEGIN
226 0959 2  local
227 0960 2  clause: ref node,
228 0961 2  item: ref node,
229 0962 2  last_item: ref node;
230 0963 2  !
231 0964 2  ! Determine which type of clause we have.
232 0965 2  if token_is(tkn_k_symbol,'KEYWORD') then (
233 0966 2  ! We have a KEYWORD definition. The first thing must be the
234 0967 2  ! keyword name. Create a node for it.
235 0968 2  cdu$get_next_token();
236 0969 2  clause = cdu$create_node(node_k_keyword,.cdu$gq_token[ptr],.cdu$gq_token[ptr]);
237 0970 2  ! This clause conflicts with any existing KEYWORD clause with the same
238 0971 2  ! name.
239 0972 2  if cdu$lookup_child(.parent,node_k_keyword,
240 0973 2  .clause[node_b_text_length],clause[node_t_text]) neq 0 then
241 0974 2  cdu$report_syntax_error(msg(cdu$dupkey),1,clause[node_b_text_length]);
242 0975 2  cdu$token_must_be(tkn_k_symbol);
243 0976 2  ! We have a list of keyword definition clauses. Each is optionally
244 0977 2  ! preceded by a comma. All of the items are chained as children of the
245 0978 2  ! main qualifier clause.
246 0979 2  loop (
247 0980 2  skip_optional_token(tkn_k_comma);
248 0981 2  item = cdu$keyword_clause(.clause);
249 0982 2  if .item eql 0 then exitloop;
250 0983 2  link_parent_to_child(clause,item,last_item);
251 0984 2  );
252 0985 2  return .clause;
253 0986 2  );
254 0987 2  if token_is(tkn_k_symbol,'PREFIX') then (
255 0988 2  ! We have an PREFIX clause. It conflicts with any existing clause.
256 0989 2  !
257 0990 2  !
258 0991 2  !
259 0992 2  !
260 0993 2  !
261 0994 2  !
262 0995 2  !
263 0996 2  !
264 0997 2  !
265 0998 2  !
266 0999 2  !
267 1000 2  !
268 1001 2  !

```

```

: 269      1002      3      if cdu$check_for_children(.parent,node_k_prefix) then
: 270      1003      3      cdu$report_syntax_error(msg(cdu$_dupprefix));
: 271      1004      3
: 272      1005      3      ! Bypass any optional equal sign.
: 273      1006      3
: 274      1007      3      cdu$get_next_token();
: 275      1008      3      skip_optional_token(tkn_k_equal);
: 276      1009      3
: 277      1010      3      ! Now we have a symbol which is the prefix. Store it in a node.
: 278      1011      3
: 279      1012      3      clause = cdu$create_node(node_k_prefix,.cdu$gq_token[len],.cdu$gq_token[ptr]);
: 280      1013      3      cdu$token_must_be(tkn_k_symbol);
: 281      1014      3      return .clause;
: 282      1015      3      );
: 283      1016      3
: 284      1017      3      ! There are no more type clauses.
: 285      1018      3
: 286      1019      3      return 0;
: 287      1020      3
: 288      1021      3      END;

```

INFO#250 L1:0992
Referenced LOCAL symbol LAST_ITEM is probably not initialized

				.PSECT \$SPLITS,NOWRT,NOEXE,2							
44	52	4F	57	59	45	4B	0003E	P.AAI:	.ASCII \KEYWORD\	:	
	58	49	46	45	52	50	00045	P.AAJ:	.ASCII \PREFIX\	:	
				.EXTRN CDU\$_DUPKEY, CDU\$_DUPPREFIX							
				.PSECT \$CODE\$,NOWRT,2							
				.ENTRY CDU\$TYPE_CLAUSE, Save R2,R3,R4,R5,R6,R7,R8,-; R9,R10				0957			
				5A	00000000G	00	9E	00002	MOVAB	CDU\$TOKEN_MUST_BE, R10	
				59	00000000G	00	9E	00009	MOVAB	CDU\$REPORT_SYNTAX_ERROR, R9	
				58	00000000G	00	9E	0C010	MOVAB	CDU\$CREATE_NODE, R8	
				57	00000000G	00	9E	00017	MOVAB	CDU\$GL_TOKEN_CLASS, R7	
				56	00000000G	00	9E	0001E	MOVAB	CDU\$GET_NEXT_TOKEN, R6	
				55	00000000G	00	9E	00025	MOVAB	CDU\$GQ_TOKEN+4, R5	
				0D		67	D1	0002C	CMPL	CDU\$GL_TOKEN_CLASS, #13	0968
						71	12	0002F	BNEQ	6\$	
				50		65	D0	00031	MOVL	CDU\$GQ_TOKEN+4, R0	
07				60	FC	A5	2D	00034	CMPCS	CDU\$GQ_TOKEN, (R0), #0, #7, P.AAI	
					0000	CF		0003A			
						63	12	0003D	BNEQ	6\$	
				66		00	FB	0003F	CALLS	#0, CDU\$GET_NEXT_TOKEN	0973
						65	DD	00042	PUSHL	CDU\$GQ_TOKEN+4	0974
				7E	FC	A5	3C	00044	MOVZWL	CDU\$GQ_TOKEN, -(SP)	
						18	DD	00048	PUSHL	#24	
				68		03	FB	0004A	CALLS	#3, CDU\$CREATE_NODE	
				54		50	D0	0004D	MOVL	R0, CLAUSE	
					11	A4	9F	00050	PUSHAB	17(CLAUSE)	0980
				7E	10	A4	9A	00053	MOVZBL	16(CLAUSE), -(SP)	
						18	DD	00057	PUSHL	#24	

00000000G	00	04	AC	DD	00059	PUSHL	PARENT	
			04	FB	0005C	CALLS	#4, CDUS\$LOOKUP_CHILD	
			50	D5	00063	TSTL	R0	
			0E	13	00065	BEQL	1\$	
		10	A4	9F	00067	PUSHAB	16(CLAUSE)	0981
			01	DD	0006A	PUSHL	#1	
	00000000G		8F	DD	0006C	PUSHL	#CDUS\$ DUPKEY	
69			03	FB	00072	CALLS	#3, CDUS\$REPORT_SYNTAX_ERROR	
			0D	DD	00075	PUSHL	#13	0982
6A			01	FB	00077	CALLS	#1, CDUSTOKEN_MUST_BE	
05			67	D1	0007A	CMPL	CDUS\$GL_TOKEN_CLASS, #5	0989
			03	12	0007D	BNEQ	3\$	
66			00	FB	0007F	CALLS	#0, CDUS\$GET_NEXT_TOKEN	
			54	DD	00082	PUSHL	CLAUSE	0990
0000V	CF		01	FB	00084	CALLS	#1, CDUS\$KEYWORD_CLAUSE	
	53		50	D0	00089	MOVL	R0, ITEM	
			5D	13	0008C	BEQL	9\$	0991
		08	A4	D5	0008E	TSTL	8(CLAUSE)	0992
			06	12	00091	BNEQ	4\$	
08	A4		53	D0	00093	MOVL	ITEM, 8(CLAUSE)	
			04	11	00097	BRB	5\$	
04	A2		53	D0	00099	MOVL	ITEM, 4(LAST_ITEM)	
	52		53	D0	0009D	MOVL	ITEM, LAST_ITEM	
			D8	11	000A0	BRB	2\$	0982
			67	D1	000A2	CMPL	CDUS\$GL_TOKEN_CLASS, #13	0998
			48	12	000A5	BNEQ	10\$	
			65	D0	000A7	MOVL	CDUS\$GQ_TOKEN+4, R0	
06			A5	2D	000AA	CMPC5	CDUS\$GQ_TOKEN, (R0), #0, #6, P.AAJ	
	00		CF		000B0			
		FC	3A	12	000B3	BNEQ	10\$	
		0000	0F	DD	000B5	PUSHL	#15	1002
			AC	DD	000B7	PUSHL	PARENT	
00000000G	00	04	02	FB	000BA	CALLS	#2, CDUS\$CHECK_FOR_CHILDREN	
	09		50	E9	000C1	BLBC	R0, 7\$	
			8F	DD	000C4	PUSHL	#CDUS\$ DUPPREFIX	1003
69			01	FB	000CA	CALLS	#1, CDUS\$REPORT_SYNTAX_ERROR	
66			00	FB	000CD	CALLS	#0, CDUS\$GET_NEXT_TOKEN	1007
06			67	D1	000D0	CMPL	CDUS\$GL_TOKEN_CLASS, #6	1008
			03	12	000D3	BNEQ	8\$	
66			00	FB	000D5	CALLS	#0, CDUS\$GET_NEXT_TOKEN	
			65	DD	000D8	PUSHL	CDUS\$GQ_TOKEN+4	1012
7E		FC	A5	3C	000DA	MOVZWL	CDUS\$GQ_TOKEN, -(SP)	
			0F	DD	000DE	PUSHL	#15	
68			03	FB	000E0	CALLS	#3, CDUS\$CREATE_NODE	
54			50	D0	000E3	MOVL	R0, CLAUSE	
			0D	DD	000E6	PUSHL	#13	1013
6A			01	FB	000E8	CALLS	#1, CDUSTOKEN_MUST_BE	
50			54	D0	000EB	MOVL	CLAUSE, R0	1014
			04		000EE	RET		
			50	D4	000EF	CLRL	R0	1019
			04		000F1	RET		1021

; Routine Size: 242 bytes, Routine Base: \$CODE\$ + 01BC

```

290 1022 1 1 ++
291 1023 1 1 Description: This syntax routine recognizes the 'keyword-clause', which
292 1024 1 1 is used to define the characteristics of a data type keyword.
293 1025 1 1
294 1026 1 1 Parameters: parent By reference, parent node of this construct.
295 1027 1 1
296 1028 1 1 Returns: The top-level node representing the clause, or zero if
297 1029 1 1 there aren't any more.
298 1030 1 1
299 1031 1 1 Notes:
300 1032 1 1 --
301 1033 1 1
302 1034 1 1 GLOBAL ROUTINE cdu$keyword_clause(parent: ref node)
303 1035 2 2 = BEGIN
304 1036 2 2
305 1037 2 2 local
306 1038 2 2 clause: ref node;
307 1039 2 2
308 1040 2 2
309 1041 2 2 ! Determine which kind of clause we have.
310 1042 2 2
311 1043 2 2 if token_is(tkn_k_symbol,'NEGATABLE') then (
312 1044 2 2
313 1045 2 2 ! We have a NEGATABLE clause. It is represented by a node.
314 1046 2 2 ! This clause conflicts with any existing NONNEGATABLE clause.
315 1047 2 2
316 1048 2 2 if cdu$check_for_children(.parent,node_k_nonnegatable) then
317 1049 2 2 cdu$report_syntax_error(msg(cdu$_confnonneg));
318 1050 2 2
319 1051 2 2 cdu$get_next_token();
320 1052 2 2 return cdu$create_node(node_k_negatable);
321 1053 2 2 );
322 1054 2 2
323 1055 2 2 if token_is(tkn_k_symbol,'NONNEGATABLE') then (
324 1056 2 2
325 1057 2 2 ! We have a NONNEGATABLE clause. It is represented by a node.
326 1058 2 2 ! This clause conflicts with any existing NEGATABLE clause.
327 1059 2 2
328 1060 2 2 if cdu$check_for_children(.parent,node_k_negatable) then
329 1061 2 2 cdu$report_syntax_error(msg(cdu$_confneg));
330 1062 2 2
331 1063 2 2 cdu$get_next_token();
332 1064 2 2 return cdu$create_node(node_k_nonnegatable);
333 1065 2 2 );
334 1066 2 2
335 1067 2 2 ! We either have a common clause now, or we've run out of clauses.
336 1068 2 2
337 1069 2 2 return cdu$common_clause(.parent);
338 1070 2 2
339 1071 1 1 END;

```

.PSECT \$SPLITS,NOWRT,NOEXE,2

```

45 4C 42 45 4C 42 41 54 41 47 45 4E 0004B P.AAK: .ASCII \NEGATABLE\
45 4C 42 41 54 41 47 45 4E 4E 4F 4E 00054 P.AAL: .ASCII \NONNEGATABLE\

```

				.PSECT	\$CODE\$,NOWRT,2	
				.ENTRY	CDUSKEYWORD_CLAUSE, Save R2,R3,R4,R5,R6,R7,-;	1034
					R8	
				MOVAB	CDUSGL_TOKEN_CLASS, R8	
				MOVAB	CDUSGET_NEXT_TOKEN, R7	
				MOVAB	CDUSREPORT_SYNTAX_ERROR, R6	
				MOVAB	CDUSCHECK_FOR_CHILDREN, R5	
				MOVAB	CDUSGQ_TOKEN+4, R4	
				CMPL	CDUSGL_TOKEN_CLASS, #13	1043
				BNEQ	2\$	
				MOVL	CDUSGQ_TOKEN+4, R0	
				CMPC5	CDUSGQ_TOKEN, (R0), #0, #9, P.AAK	
				BNEQ	2\$	
				PUSHL	#22	1048
				PUSHL	PARENT	
				CALLS	#2, CDUSCHECK_FOR_CHILDREN	
				BLBC	R0, 1\$	
				PUSHL	#CDUS_CONFNONNEG	1049
				CALLS	#1, CDUSREPORT_SYNTAX_ERROR	
				CALLS	#0, CDUSGET_NEXT_TOKEN	1051
				PUSHL	#21	1052
				BRB	4\$	
				CMPL	CDUSGL_TOKEN_CLASS, #13	1055
				BNEQ	5\$	
				MOVL	CDUSGQ_TOKEN+4, R0	
				CMPC5	CDUSGQ_TOKEN, (R0), #0, #12, P.AAL	
				BNEQ	5\$	
				PUSHL	#21	1060
				PUSHL	PARENT	
				CALLS	#2, CDUSCHECK_FOR_CHILDREN	
				BLBC	R0, 3\$	
				PUSHL	#CDUS_CONFNEG	1061
				CALLS	#1, CDUSREPORT_SYNTAX_ERROR	
				CALLS	#0, CDUSGET_NEXT_TOKEN	1063
				PUSHL	#22	1064
				CALLS	#1, CDUSCREATE_NODE	
				RET		
				PUSHL	PARENT	1069
				CALLS	#1, CDUSCOMMON_CLAUSE	
				RET		1071

; Routine Size: 144 bytes, Routine Base: \$CODE\$ + 02AE

```

341 1072 1 |++
342 1073 1 |Description: This syntax routine recognizes the "common-clause" construct,
343 1074 1 |which is used to define the characteristics of parameters,
344 1075 1 |qualifiers, and keywords.
345 1076 1 |
346 1077 1 |Parameters: None
347 1078 1 |
348 1079 1 |Returns: The top-level node representing the clause, or zero if
349 1080 1 |there aren't any more.
350 1081 1 |
351 1082 1 |Notes:
352 1083 1 |--
353 1084 1 |
354 1085 1 |GLOBAL ROUTINE cdu$common_clause(parent: ref node)
355 1086 2 |= BEGIN
356 1087 2 |
357 1088 2 |local
358 1089 2 |    clause: ref node,
359 1090 2 |    item: ref node,
360 1091 2 |    last_item: ref node;
361 1092 2 |
362 1093 2 |
363 1094 2 |! Determine which kind of clause we have.
364 1095 2 |
365 1096 2 |if token_is(tkn_k_symbol,'CLIFLAGS') then (
366 1097 2 |
367 1098 2 |    ! We have a CLIFLAGS clause. Create a parent node for the flags.
368 1099 2 |
369 1100 2 |    clause = cdu$create_node(node_k_cliflags);
370 1101 2 |    cdu$get_next_token();
371 1102 2 |
372 1103 2 |    ! We have a parenthesized list of CLI flags.
373 1104 2 |    ! Eat the open parenthesis. Then sit in a loop which recognizes at
374 1105 2 |    ! least one item, along with any others separated by commas. Finally,
375 1106 2 |    ! eat the close parenthesis. All of the items are chained as children
376 1107 2 |    ! of the clause.
377 1108 2 |
378 1109 2 |    cdu$token_must_be(tkn_k_open_paren);
379 1110 2 |    loop (
380 1111 2 |        item = cdu$cli_flag();
381 1112 2 |        link_parent_to_child(clause,item,last_item);
382 1113 2 |        if not token_is(tkn_k_comma) then exit loop;
383 1114 2 |        cdu$get_next_token();
384 1115 2 |    );
385 1116 2 |    cdu$token_must_be(tkn_k_close_paren);
386 1117 2 |    return .clause;
387 1118 2 |);
388 1119 2 |
389 1120 2 |if token_is(tkn_k_symbol,'DEFAULT') then (
390 1121 2 |
391 1122 2 |    ! We have a DEFAULT clause. It is represented by a node.
392 1123 2 |
393 1124 2 |    cdu$get_next_token();
394 1125 2 |    return cdu$create_node(node_k_default);
395 1126 2 |);
396 1127 2 |
397 1128 2 |if token_is(tkn_k_symbol,'LABEL') then (

```

```

398      1129      3
399      1130      3
400      1131      3
401      1132      3
402      1133      3
403      1134      3
404      1135      3
405      1136      3
406      1137      3
407      1138      3
408      1139      3
409      1140      3
410      1141      3
411      1142      3
412      1143      3
413      1144      3
414      1145      3
415      1146      3
416      1147      3
417      1148      3
418      1149      3
419      1150      3
420      1151      3
421      1152      3
422      1153      3
423      1154      3
424      1155      3
425      1156      3
426      1157      3
427      1158      3
428      1159      3
429      1160      3
430      1161      3
431      1162      3
432      1163      3
433      1164      3
434      1165      3
435      1166      3
436      1167      3
437      1168      3
438      1169      3
439      1170      3
440      1171      3
441      1172      3
442      1173      3
443      1174      3
444      1175      3
445      1176      3
446      1177      3
447      1178      3
448      1179      3
449      1180      3
450      1181      3
451      1182      3
452      1183      3
453      1184      3
454      1185      4

! We have an LABEL clause. It conflicts with any existing clause.
if cdu$check_for_children(.parent,node_k_label) then
    cdu$report_syntax_error(msg(cdu$_duplabel));

! Bypass any optional equal sign.
cdu$get_next_token();
skip_optional_token(tkn_k_equal);

! Finally we have a symbol which is the label. Make a node for it.
clause = cdu$create_node(node_k_label,.cdu$gq_token[len],.cdu$gq_token[ptr]);
cdu$token_must_be(tkn_k_symbol);
return .clause;
);

if token_is(tkn_k_symbol,'SYNTAX') then (
    ! We have an SYNTAX clause. It conflicts with any existing clause.
    if cdu$check_for_children(.parent,node_k_syntax) then
        cdu$report_syntax_error(msg(cdu$_dupsyntax));

    ! Bypass any optional equal sign.
    cdu$get_next_token();
    skip_optional_token(tkn_k_equal);

    ! Finally we have a symbol which is the name of the syntax definition.
    ! Make a node for it.
    clause = cdu$create_node(node_k_syntax,.cdu$gq_token[len],.cdu$gq_token[ptr]);
    cdu$token_must_be(tkn_k_symbol);
    return .clause;
);

if token_is(tkn_k_symbol,'VALUE') then (
    ! We have a VALUE clause. It conflicts with any existing clause.
    if cdu$check_for_children(.parent,node_k_value) then
        cdu$report_syntax_error(msg(cdu$_dupvalue));

    ! Create a node to represent the clause.
    clause = cdu$create_node(node_k_value);
    cdu$get_next_token();

    ! We have an optional parenthesized list of subclauses.
    ! Eat the open parenthesis. Then sit in a loop which recognizes at
    ! least one item, along with any others separated by commas. Finally,
    ! eat the close parenthesis. All of the items are chained as children
    ! of the clause.

    if token_is(tkn_k_open_paren) then (

```

```

: 455      1186  4      cdu$get_next_token();
: 456      1187  5      loop (
: 457      1188  5          item = cdu$value clause(.clause);
: 458      1189  5          link_parent_to_child(clause,item,last item);
: 459      1190  5          if not token_is(tkn_k_comma) then exitloop;
: 460      1191  5          cdu$get_next_token();
: 461      1192  4      );
: 462      1193  4      cdu$token_must_be(tkn_k_close_paren);
: 463      1194  3      );
: 464      1195  3      return .clause;
: 465      1196  2      );
: 466      1197  2      ! We don't have any more common clauses.
: 467      1198  2
: 468      1199  2      return 0;
: 469      1200  2
: 470      1201  2      return 0;
: 471      1202  2
: 472      1203  1      END;
: INFO#250      LI:1112
: Referenced LOCAL symbol LAST_ITEM is probably not initialized

```

```

.PSECT $SPLITS$,NOWRT,NOEXE,2
53 47 41 4C 46 49 4C 43 00060 P.AAM: .ASCII \CLIFLAGS\
54 4C 55 41 46 45 44 00068 P.AAN: .ASCII \DEFAULT\
      4C 45 42 41 4C 0006F P.AAO: .ASCII \LABEL\
      58 41 54 4E 59 53 00074 P.AAP: .ASCII \SYNTAX\
      45 55 4C 41 56 0007A P.AAQ: .ASCII \VALUE\

.EXTRN CDU$_DUPLABEL, CDU$_DUPSYNTAX
.EXTRN CDU$_DUPVALUE

.PSECT $CODES$,NOWRT,2
.OFFC 00000
.ENTRY CDU$COMMON CLAUSE, Save R2,R3,R4,R5,R6,R7,- 1085
      R8,R9,R10,R11
      MOVAB CDU$CHECK_FOR_CHILDREN, R11
      MOVAB CDU$CREATE_NODE, R10
      MOVAB CDU$GET_NEXT_TOKEN, R9
      MOVAB CDU$GL_TOKEN_CLASS, R8
      MOVAB CDU$GQ_TOKEN+4, R7
      CMPL CDU$GL_TOKEN_CLASS, #13 1096
      BNEQ 5$
      MOVL CDU$GQ_TOKEN+4, R0
      CMPC5 CDU$GQ_TOKEN, (R0), #0, #8, P.AAM
      BNEQ 5$
      PUSHL #7 1100
      CALLS #1, CDU$CREATE_NODE
      MOVL R0, CLAUSE
      CALLS #0, CDU$GET_NEXT_TOKEN 1101
      PUSHL #7 1109
      CALLS #1, CDU$TOKEN_MUST_BE
      CALLS #0, CDU$CLI_FLAG 1111
      MOVL R0, ITEM

```

		08	A4	D5	00054	TSTL	8(CLAUSE)	1112
			06	12	00057	BNEQ	2\$	
08	A4		56	D0	00059	MOVL	ITEM, 8(CLAUSE)	
			04	11	0005D	BRB	3\$	
04	A5		56	D0	0005F	2\$: MOVL	ITEM, 4(LAST_ITEM)	
	55		56	D0	00063	3\$: MOVL	ITEM, LAST_ITEM	
	05		68	D1	00066	CMPL	CDUSGL_TOKEN_CLASS, #5	1113
			03	13	00069	BEQL	4\$	
		0108	31	0006B	BRW	18\$		
	69		00	FB	0006E	4\$: CALLS	#0, CDUSGET_NEXT_TOKEN	1114
			D9	11	00071	BRB	1\$	1109
	0D		68	D1	00073	5\$: CMPL	CDUSGL_TOKEN_CLASS, #13	1120
			17	12	00076	BNEQ	6\$	
	50		67	D0	00078	MOVL	CDUSGQ_TOKEN+4, R0	
07	00		60	A7	2D	CMPC5	CDUSGQ_TOKEN, (R0), #0, #7, P.AAN	
		FC		CF	00081			
		0000		09	12	BNEQ	6\$	
			69	00	FB	CALLS	#0, CDUSGET_NEXT_TOKEN	1124
				19	DD	PUSHL	#25	1125
			6A	01	FB	CALLS	#1, CDUSCREATE_NODE	
				04	0008E	RET		
	0D		68	D1	0008F	6\$: CMPL	CDUSGL_TOKEN_CLASS, #13	1128
			3B	12	00092	BNEQ	9\$	
	50		67	D0	00094	MOVL	CDUSGQ_TOKEN+4, R0	
05	00		60	A7	2D	CMPC5	CDUSGQ_TOKEN, (R0), #0, #5, P.AAO	
		FC		CF	0009D			
		0000		2D	12	BNEQ	9\$	
			1A	DD	000A2	PUSHL	#26	1132
		04	AC	DD	000A4	PUSHL	PARENT	
	6B		02	FB	000A7	CALLS	#2, CDUSCHECK_FOR_CHILDREN	
	0D		50	E9	000AA	BLBC	R0, 7\$	
		00000000G	8F	DD	000AD	PUSHL	#CDUS DUPLABEL	1133
	00		01	FB	000B3	CALLS	#1, CDUSREPORT SYNTAX ERROR	
	69		00	FB	000BA	7\$: CALLS	#0, CDUSGET NEXT TOKEN	1137
	06		68	D1	000BD	CMPL	CDUSGL_TOKEN_CLASS, #6	1138
			03	12	000C0	BNEQ	8\$	
	69		00	FB	000C2	CALLS	#0, CDUSGET NEXT_TOKEN	
			67	DD	000C5	8\$: PUSHL	CDUSGQ_TOKEN+4	1142
	7E		A7	3C	000C7	MOVZWL	CDUSGQ_TOKEN, -(SP)	
		FC	1A	DD	000CB	PUSHL	#26	
			3E	11	000CD	BRB	12\$	
	0D		68	D1	000CF	9\$: CMPL	CDUSGL_TOKEN_CLASS, #13	1147
			43	12	000D2	BNEQ	13\$	
	50		67	D0	000D4	MOVL	CDUSGQ_TOKEN+4, R0	
06	00		60	A7	2D	CMPC5	CDUSGQ_TOKEN, (R0), #0, #6, P.AAP	
		FC		CF	000DD			
		0000		35	12	BNEQ	13\$	
			1B	DD	000E2	PUSHL	#27	1151
		04	AC	DD	000E4	PUSHL	PARENT	
	6B		02	FB	000E7	CALLS	#2, CDUSCHECK_FOR_CHILDREN	
	0D		50	E9	000EA	BLBC	R0, 10\$	
		00000000G	8F	DD	000ED	PUSHL	#CDUS DUPSYNTAX	1152
	00		01	FB	000F3	CALLS	#1, CDUSREPORT SYNTAX ERROR	
	69		00	FB	000FA	10\$: CALLS	#0, CDUSGET NEXT TOKEN	1156
	06		68	D1	000FD	CMPL	CDUSGL_TOKEN_CLASS, #6	1157
			03	12	00100	BNEQ	11\$	
	69		00	FB	00102	CALLS	#0, CDUSGET_NEXT_TOKEN	

			67	DD	00105	11\$:	PUSHL	CDUSGQ_TOKEN+4	1162
	7E	FC	A7	3C	00107		MOVZWL	CDUSGQ_TOKEN, -(SP)	
			1B	DD	0010B		PUSHL	#27	
	6A		03	FB	0010D	12\$:	CALLS	#3, CDUSCREATE_NODE	
	54		50	D0	00110		MOVL	R0, CLAUSE	
			0D	DD	00113		PUSHL	#13	1163
			61	11	00115		BRB	19\$	
	0D		68	D1	00117	13\$:	CMPL	CDUSGL_TOKEN_CLASS, #13	1167
			67	12	0011A		BNEQ	21\$	
	50		67	D0	0011C		MOVL	CDUSGQ_TOKEN+4, R0	
05	00		60	A7	2D	0011F	CMPC5	CDUSGQ_TOKEN, (R0), #0, #5, P.AAQ	
		FC	CF		00125				
		0000	59	12	00128		BNEQ	21\$	
			1C	DD	0012A		PUSHL	#28	1171
		04	AC	DD	0012C		PUSHL	PARENT	
	6B		02	FB	0012F		CALLS	#2, CDUSCHECK_FOR_CHILDREN	
	0D		50	E9	00132		BLBC	R0, 14\$	
		00000000G	8F	DD	00135		PUSHL	#CDUS_DUPVALUE	1172
		00	01	FB	0013B		CALLS	#1, CDUSREPORT_SYNTAX_ERROR	
			1C	DD	00142	14\$:	PUSHL	#28	1176
	6A		01	FB	00144		CALLS	#1, CDUSCREATE_NODE	
	54		50	D0	00147		MOVL	R0, CLAUSE	
	69		00	FB	0014A		CALLS	#0, CDUSGET_NEXT_TOKEN	1177
	07		68	D1	0014D		CMPL	CDUSGL_TOKEN_CLASS, #7	1185
			2D	12	00150		BNEQ	20\$	
	69		00	FB	00152	15\$:	CALLS	#0, CDUSGET_NEXT_TOKEN	1186
			54	DD	00155		PUSHL	CLAUSE	1188
	0000V	CF	01	FB	00157		CALLS	#1, CDUSVALUE_CLAUSE	
	56		50	D0	0015C		MOVL	R0, ITEM	
		08	A4	D5	0015F		TSTL	8(CLAUSE)	1189
			06	12	00162		BNEQ	16\$	
	08	A4	56	D0	00164		MOVL	ITEM, 8(CLAUSE)	
			04	11	00168		BRB	17\$	
	04	A5	56	D0	0016A	16\$:	MOVL	ITEM, 4(LAST_ITEM)	
	55		56	D0	0016E	17\$:	MOVL	ITEM, LAST_ITEM	
	05		68	D1	00171		CMPL	CDUSGL_TOKEN_CLASS, #5	1190
			DC	13	00174		BEQL	15\$	
			08	DD	00176	18\$:	PUSHL	#8	1193
	00000000G	00	01	FB	00178	19\$:	CALLS	#1, CDUSTOKEN_MUST_BE	
		50	54	D0	0017F	20\$:	MOVL	CLAUSE, R0	1196
				04	00182		RET		
			50	D4	00183	21\$:	CLRL	R0	1201
			04		00185		RET		1203

; Routine Size: 390 bytes, Routine Base: \$CODE\$ + 033E

```

474 1204 1 1 ++
475 1205 1 1 Description: This syntax routine recognizes a 'value-clause' construct,
476 1206 1 1 which is used to define the characteristics of a value.
477 1207 1 1
478 1208 1 1 Parameters: parent By reference, parent node of this construct.
479 1209 1 1
480 1210 1 1 Returns: The top-level node for the value clause.
481 1211 1 1
482 1212 1 1 Notes:
483 1213 1 1 --
484 1214 1 1
485 1215 1 1 GLOBAL ROUTINE cdu$value_clause(parent: ref node)
486 1216 2 2 = BEGIN
487 1217 2 2
488 1218 2 2 local
489 1219 2 2 clause: ref node;
490 1220 2 2
491 1221 2 2
492 1222 2 2 ! Determine which kind of clause we have.
493 1223 2 2
494 1224 2 2 if token_is(tkn_k_symbol,'CONCATENATE') then (
495 1225 2 2
496 1226 2 2 ! We have a CONCATENATE clause. It conflicts with any existing
497 1227 2 2 ! NOCONCATENATE clause.
498 1228 2 2
499 1229 2 2 if cdu$check_for_children(.parent,node_k_noconcatenate) then
500 1230 2 2 cdu$report_syntax_error(msg(cdu$_confnocon));
501 1231 2 2
502 1232 2 2 ! Represent the clause with a node.
503 1233 2 2
504 1234 2 2 cdu$get_next_token();
505 1235 2 2 return cdu$create_node(node_k_concatenate);
506 1236 2 2 );
507 1237 2 2
508 1238 2 2 if token_is(tkn_k_symbol,'NOCONCATENATE') then (
509 1239 2 2
510 1240 2 2 ! We have a NOCONCATENATE clause. It conflicts with any existing
511 1241 2 2 ! CONCATENATE clause.
512 1242 2 2
513 1243 2 2 if cdu$check_for_children(.parent,node_k_concatenate) then
514 1244 2 2 cdu$report_syntax_error(msg(cdu$_confcon));
515 1245 2 2
516 1246 2 2 ! Represent the clause with a node.
517 1247 2 2
518 1248 2 2 cdu$get_next_token();
519 1249 2 2 return cdu$create_node(node_k_noconcatenate);
520 1250 2 2 );
521 1251 2 2
522 1252 2 2 if token_is(tkn_k_symbol,'DEFAULT') then (
523 1253 2 2
524 1254 2 2 ! We have a DEFAULT clause. It conflicts with any existing DEFAULT
525 1255 2 2 ! or REQUIRED clause.
526 1256 2 2
527 1257 2 2 if cdu$check_for_children(.parent,node_k_default,node_k_required) then
528 1258 2 2 cdu$report_syntax_error(msg(cdu$_confdefreq));
529 1259 2 2
530 1260 2 2 ! Bypass any optional equal sign.

```

```

: 531      1261
: 532      1262
: 533      1263
: 534      1264
: 535      1265
: 536      1266
: 537      1267
: 538      1268
: 539      P 1269
: 540      P 1270
: 541      P 1271
: 542      P 1272
: 543      P 1273
: 544      P 1274
: 545      P 1275
: 546      P 1276
: 547      P 1277
: 548      P 1278
: 549      P 1279
: 550      P 1280
: 551      P 1281
: 552      P 1282
: 553      P 1283
: 554      P 1284
: 555      P 1285
: 556      P 1286
: 557      P 1287
: 558      P 1288
: 559      P 1289
: 560      P 1290
: 561      P 1291
: 562      P 1292
: 563      P 1293
: 564      P 1294
: 565      P 1295
: 566      P 1296
: 567      P 1297
: 568      P 1298
: 569      P 1299
: 570      P 1300
: 571      P 1301
: 572      P 1302
: 573      P 1303
: 574      P 1304
: 575      P 1305
: 576      P 1306
: 577      P 1307
: 578      P 1308
: 579      P 1309
: 580      P 1310
: 581      1311
: 582      1312
: 583      1313
: 584      1314
: 585      1315
: 586      1316
: 587      1317

```

```

cdu$get_next_token(tkn_k_h_string);
skip_optional_token(tkn_k_equal,tkn_k_h_string);

! The next token is either the default value, or an open
! parenthesis to start a list of values. We need a buffer to
! accumulate the values.
with_dbuffer(values,tkn_k_max_length,

! Clear the buffer.
values[len] = 0;

if not token_is(tkn_k_open_paren) then (

! It is not an open parenthesis, so it must be the
! default value. Add it to our buffer as an ASCII
! string.

ch$wchar(.cdu$gq_token[len],.values[ptr]);
ch$move(.cdu$gq_token[len],.cdu$gq_token[ptr],.values[ptr]+1);
values[len] = 1 + .cdu$gq_token[len];
cdu$token_must_be(tkn_k_string);

) else (

! We have a parenthesized list of default values.
! Eat the open parenthesis. Then sit in a loop which
! recognizes at least one value, along with any others
! separated by commas. Finally, eat the close paren.
! All of the names go in the buffer as ASCII strings.

cdu$get_next_token();
loop (
    ch$wchar(.cdu$gq_token[len],.values[ptr]+.values[len]);
    ch$move(.cdu$gq_token[len],.cdu$gq_token[ptr],
        .values[ptr]+.values[len]+1);
    values[len] = .values[len] + 1 + .cdu$gq_token[len];
    cdu$token_must_be(tkn_k_string);
    if not token_is(tkn_k_comma) then exitloop;
    cdu$get_next_token();
);
cdu$token_must_be(tkn_k_close_paren);

);

! Create a node to represent the defaults, and put the values
! in it.
clause = cdu$create_node(node_k_default,.values[len],.values[ptr]);

);

! Check the length of the default values..
if .clause[node_b_text_length] gtr ent_k_max_defval-1 then
    cdu$report_syntax_error(msg(cdu$_defaultlen),1,ent_k_max_defval-2);

```

```

588 1318 3      return .clause;
589 1319 3      );
590 1320 3
591 1321 3      if token_is(tkn_k_symbol,'IMPCAT') then (
592 1322 3          ! We have an IMPCAT clause. It is represented by a node.
593 1323 3
594 1324 3          cdu$get_next_token();
595 1325 3          return cdu$create_node(node_k_impcat);
596 1326 3
597 1327 3      );
598 1328 3
599 1329 3      if token_is(tkn_k_symbol,'LIST') then (
600 1330 3          ! We have a LIST clause. It is represented by a node.
601 1331 3
602 1332 3          cdu$get_next_token();
603 1333 3          return cdu$create_node(node_k_list);
604 1334 3
605 1335 3      );
606 1336 3
607 1337 3      if token_is(tkn_k_symbol,'REQUIRED') then (
608 1338 3          ! We have a REQUIRED clause. It is represented by a node.
609 1339 3          ! This clause conflicts with any existing DEFAULT clause.
610 1340 3
611 1341 3          if cdu$check_for_children(.parent,node_k_default) then
612 1342 3              cdu$report_syntax_error(msg(cdu$_confdefreq));
613 1343 3
614 1344 3          cdu$get_next_token();
615 1345 3          return cdu$create_node(node_k_required);
616 1346 3
617 1347 3      );
618 1348 3
619 1349 3      if token_is(tkn_k_symbol,'TYPE') then (
620 1350 3          ! We have a TYPE clause. It conflicts with any existing clause.
621 1351 3
622 1352 3          if cdu$check_for_children(.parent,node_k_type_builtin,node_k_type_user) then
623 1353 3              cdu$report_syntax_error(msg(cdu$_duptype));
624 1354 3
625 1355 3          ! Bypass any optional equal sign.
626 1356 3
627 1357 3          cdu$get_next_token();
628 1358 3          skip_optional_token(tkn_k_equal);
629 1359 3
630 1360 3          ! Now we have a symbol which is the name of the desired type.
631 1361 3          ! If it begins with a dollar sign, then it's a builtin type and
632 1362 3          ! we create a node with the type code. Otherwise it's a
633 1363 3          ! user-defined type and we create a node with the symbol.
634 1364 3
635 1365 3          if ch$rchar(.cdu$gq_token[ptr]) eqv '$' then (
636 1366 3              local
637 1367 3                  type_code: byte;
638 1368 3
639 1369 3                  type_code = cdu$builtin_type();
640 1370 3                  clause = cdu$create_node(node_k_type_builtin,1,type_code);
641 1371 3          ) else
642 1372 3              clause = cdu$create_node(node_k_type_user,.cdu$gq_token[len],.cdu$gq_token[ptr]);
643 1373 3
644 1374 3

```

```

: 645      1375      3      cdu$token_must_be(tkn_k_symbol);
: 646      1376      3      return .clause;
: 647      1377      2      );
: 648      1378      2
: 649      1379      2      ! Oops, it isn't one of the above clauses. Tell the user and return
: 650      1380      2      ! an error node.
: 651      1381      2
: 652      1382      2      cdu$report_syntax_error(msg(cdu$_invvalclause));
: 653      1383      2      return cdu$create_node(node_k_error);
: 654      1384      2
: 655      1385      1      END;

```

										.PSECT \$SPLITS,NOWRT,NOEXE,2							
45	54	45	54	41	4E	45	54	41	43	4E	4F	43	0007F	P.AAR:	.ASCII	\CONCATENATE\	:
		41	4E	45	54	41	43	4E	4F	43	4F	4E	0008A	P.AAS:	.ASCII	\NOCONCATENATE\	:
						54	4C	55	41	46	45	44	00097	P.AAT:	.ASCII	\DEFAULT\	:
							54	41	43	50	4D	49	0009E	P.AAU:	.ASCII	\IMPCAT\	:
									54	53	49	4C	000A4	P.AAV:	.ASCII	\LIST\	:
		44	45	52	49				55	51	45	52	000A8	P.AAW:	.ASCII	\REQUIRED\	:
									45	50	59	54	000B0	P.AAX:	.ASCII	\TYPE\	:
										.EXTRN CDU\$_CONFNOCON, CDU\$_CONFCON							
										.EXTRN CDU\$_CONFDEFREQ							
										.EXTRN CDU\$_DEFAULTLEN							
										.EXTRN CDU\$_DUPTYPE, CDU\$_INVVALCLAUSE							
										.PSECT \$CODE\$,NOWRT,2							
										.ENTRY CDU\$VALUE_CLAUSE, Save R2,R3,R4,R5,R6,R7,-		1215					
										R8,R9,R10,R11		:					
										MOVAB CDU\$REPORT_SYNTAX_ERROR, R11		:					
										MOVAB CDU\$GET_NEXT_TOKEN, R10		:					
										MOVAB CDU\$GL_TOKEN_CLASS, R9		:					
										MOVAB CDU\$GQ_TOKEN+4, R8		:					
										MOVAB -268(SP), SP		:					
										CMPL CDU\$GL_TOKEN_CLASS, #13		1224					
										BNEQ 2\$		:					
0B						50			68	D0	00028			MOVAB	CDU\$GQ_TOKEN+4, R0	:	
		00				60			FC	A8	2D	0002B		CMPC5	CDU\$GQ_TOKEN, (R0), #0, #11, P.AAR	:	
									0000	CF		00031				:	
										1F	12	00034		BNEQ	2\$	:	
										33	DD	00036		PUSHL	#51	1229	
										AC	DD	00038		PUSHL	PARENT	:	
						00000000G	00			02	FB	0003B		CALLS	#2, CDU\$CHECK_FOR_CHILDREN	:	
							09			50	E9	00042		BLBC	R0, 1\$	:	
										8F	DD	00045		PUSHL	#CDU\$_CONFNOCON	1230	
						6B				01	FB	0004B		CALLS	#1, CDU\$REPORT_SYNTAX_ERROR	:	
						6A				00	FB	0004E	1\$:	CALLS	#0, CDU\$GET_NEXT_TOKEN	1234	
										32	DD	00051		PUSHL	#50	1235	
										30	11	00053		BRB	4\$	:	
						0D				69	D1	00055	2\$:	CMPL	CDU\$GL_TOKEN_CLASS, #13	1238	
										2E	12	00058		BNEQ	5\$	:	
						50				68	D0	0005A		MOVAB	CDU\$GQ_TOKEN+4, R0	:	
0D		00				60			FC	A8	2D	0005D		CMPC5	CDU\$GQ_TOKEN, (R0), #0, #13, P.AAS	:	

			08	DD	00127		PUSHL	#8		
	00000000G	00	01	FB	00129	11\$:	CALLS	#1, CDUSTOKEN_MUST_BE		
		7E	08	AE	DD 00130		PUSHL	VALUES+4		
			08	AE	3C 00133		MOVZWL	VALUES, -(SP)		
	00000000G	00	19	DD	00137		PUSHL	#25		
	54		03	FB	00139		CALLS	#3, CDUS\$CREATE_NODE		
	SF	8F	50	DD	00140		MOVL	R0, CLAUSE		
			10	A4	91 00143		CMPB	16(CLAUSE), #95		1315
		7E	0F	1B	00148		BLEQU	12\$		
			5E	8F	9A 0014A		MOVZBL	#94, -(SP)		1316
			01	DD	0014E		PUSHL	#1		
		00000000G	8F	DD	00150		PUSHL	#CDUS\$ DEFAULTLEN		
	6B		03	FB	00156		CALLS	#3, CDUS\$REPORT_SYNTAX_ERROR		
			00D1	31	00159	12\$:	BRW	23\$		1318
	0D		69	D1	0015C	13\$:	CMPL	CDUSGL_TOKEN_CLASS, #13		1321
			15	12	0015F		BNEQ	14\$		
06	00		50	DD	00161		MOVL	CDUSGQ_TOKEN+4, R0		
			60	FC	A8 2D 00164		CMPC5	CDUSGQ_TOKEN, (R0), #0, #6, P.AAU		
				0000	CF					
			07	12	0016D		BNEQ	14\$		
	6A		00	FB	0016F		CALLS	#0, CDUS\$GET_NEXT_TOKEN		1325
			1D	DD	00172		PUSHL	#29		1326
			4A	11	00174		BRB	17\$		
	0D		69	D1	00176	14\$:	CMPL	CDUSGL_TOKEN_CLASS, #13		1329
			15	12	00179		BNEQ	15\$		
04	00		50	DD	0017B		MOVL	CDUSGQ_TOKEN+4, R0		
			60	FC	A8 2D 0017E		CMPC5	CDUSGQ_TOKEN, (R0), #0, #4, P.AAV		
				0000	CF					
			07	12	00187		BNEQ	15\$		
	6A		00	FB	00189		CALLS	#0, CDUS\$GET_NEXT_TOKEN		1333
			1E	DD	0018C		PUSHL	#30		1334
			30	11	0018E		BRB	17\$		
	0D		69	D1	00190	15\$:	CMPL	CDUSGL_TOKEN_CLASS, #13		1337
			2D	12	00193		BNEQ	18\$		
08	00		50	DD	00195		MOVL	CDUSGQ_TOKEN+4, R0		
			60	FC	A8 2D 00198		CMPC5	CDUSGQ_TOKEN, (R0), #0, #8, P.AAW		
				0000	CF					
			1F	12	001A1		BNEQ	18\$		
			19	DD	001A3		PUSHL	#25		1342
		04	AC	DD	001A5		PUSHL	PARENT		
	00000000G	00	02	FB	001A8		CALLS	#2, CDUS\$CHECK_FOR_CHILDREN		
		09	50	E9	001AF		BLBC	R0, 16\$		
		00000000G	8F	DD	001B2		PUSHL	#CDUS\$ CONFDEFREQ		1343
	6B		01	FB	001B8		CALLS	#1, CDUS\$REPORT SYNTAX ERROR		
	6A		00	FB	001BB	16\$:	CALLS	#0, CDUS\$GET_NEXT_TOKEN		1345
			1F	DD	001BE		PUSHL	#31		1346
			7A	11	001C0	17\$:	BRB	25\$		
	0D		69	D1	001C2	18\$:	CMPL	CDUSGL_TOKEN_CLASS, #13		1349
			6A	12	001C5		BNEQ	24\$		
04	00		50	DD	001C7		MOVL	CDUSGQ_TOKEN+4, R0		
			60	FC	A8 2D 001CA		CMPC5	CDUSGQ_TOKEN, (R0), #0, #4, P.AAX		
				0000	CF					
			5C	12	001D3		BNEQ	24\$		
			21	DD	001D5		PUSHL	#33		1353
			20	DD	001D7		PUSHL	#32		
		04	AC	DD	001D9		PUSHL	PARENT		
	00000000G	00	03	FB	001DC		CALLS	#3, CDUS\$CHECK_FOR_CHILDREN		

09		50	E9	001E3	BLBC	R0, 19\$	
	00000000G	8F	DD	001E6	PUSHL	#CDU\$ DUPTYPE	1354
6B		01	FB	001EC	CALLS	#1, CDU\$REPORT_SYNTAX_ERROR	
6A		00	FB	001EF	CALLS	#0, CDU\$GET_NEXT_TOKEN	1358
06		69	D1	001F2	CMPL	CDU\$GL_TOKEN_CLASS, #6	1359
		03	12	001F5	BNEQ	20\$	
6A		00	FB	001F7	CALLS	#0, CDU\$GET_NEXT_TOKEN	
50		68	D0	001FA	MOVL	CDU\$GQ_TOKEN+4, R0	1366
24		60	91	001FD	CMPB	(R0), #36	
		10	12	00200	BNEQ	21\$	
0000V	CF	00	FB	00202	CALLS	#0, CDU\$BUILTIN_TYPE	1370
	6E	50	90	00207	MOVB	R0, TYPE_CODE	
		5E	DD	0020A	PUSHL	SP	1371
		01	DD	0020C	PUSHL	#1	
		20	DD	0020E	PUSHL	#32	
		08	11	00210	BRB	22\$	
		50	DD	00212	PUSHL	R0	1373
7E	FC	A8	3C	00214	MOVZWL	CDU\$GQ_TOKEN, -(SP)	
		21	DD	00218	PUSHL	#33	
00000000G	00	03	FB	0021A	CALLS	#3, CDU\$CREATE_NODE	
	54	50	D0	00221	MOVL	R0, CLAUSE	
		0D	DD	00224	PUSHL	#13	1375
00000000G	00	01	FB	00226	CALLS	#1, CDU\$TOKEN_MUST_BE	
	50	54	D0	0022D	MOVL	CLAUSE, R0	1376
		04	00230	RET			
	00000000G	8F	DD	00231	PUSHL	#CDU\$ INVVALCLAUSE	1382
	6B	01	FB	00237	CALLS	#1, CDU\$REPORT_SYNTAX_ERROR	
		7E	D4	0023A	CLRL	-(SP)	1383
00000000G	00	01	FB	0023C	CALLS	#1, CDU\$CREATE_NODE	
		04	00243	RET			1385

; Routine Size: 580 bytes, Routine Base: \$CODE\$ + 04C4

```

657 1386 1 1 ++
658 1387 1 1 Description: This syntax routine recognizes a builtin type symbol,
659 1388 1 1 which can be specified in a TYPE clause.
660 1389 1 1
661 1390 1 1 Parameters: None.
662 1391 1 1
663 1392 1 1 Returns: By value, the corresponding numeric code. Zero is returned
664 1393 1 1 if the name is invalid.
665 1394 1 1
666 1395 1 1 Notes:
667 1396 1 1 --
668 1397 1 1
669 1398 1 1 ROUTINE cdu$builtin_type
670 1399 2 2 = BEGIN
671 1400 2 2
672 1401 2 2 own
673 1402 2 2 type_names: vector[ent_k_max_valtype+1,long] initial(
674 1403 2 2 ctext(''),
675 1404 2 2 ctext('$INFILE'),
676 1405 2 2 ctext('$OUTFILE'),
677 1406 2 2 ctext('$NUMBER'),
678 1407 2 2 ctext('$PRIVILEGE'),
679 1408 2 2 ctext('$DATETIME'),
680 1409 2 2 ctext('$PROTECTION'),
681 1410 2 2 ctext('$PROCESS'),
682 1411 2 2 ctext('$INLOG'),
683 1412 2 2 ctext('$OUTLOG'),
684 1413 2 2 ctext('$INSYM'),
685 1414 2 2 ctext('$OUTSYM'),
686 1415 2 2 ctext('$NODE'),
687 1416 2 2 ctext('$DEVICE'),
688 1417 2 2 ctext('$DIRECTORY'),
689 1418 2 2 ctext('$UIC'),
690 1419 2 2 ctext('$REST OF LINE'),
691 1420 2 2 ctext('$PARENTHESESIZED_VALUE'),
692 1421 2 2 ctext('$DELTATIME'),
693 1422 2 2 ctext('$QUOTED_STRING'),
694 1423 2 2 ctext('$FILE'),
695 1424 2 2 ctext('$EXPRESSION'),
696 1425 2 2 ctext('$TEST1'),
697 1426 2 2 ctext('$TEST2'),
698 1427 2 2 ctext('$TEST3'),
699 1428 2 2 ctext('$ACL')
700 1429 2 2 ),
701 1430 2 2
702 1431 2 2 type_codes: vector[ent_k_max_valtype+1,byte] initial(byte(
703 1432 2 2 0,
704 1433 2 2 ent_k_infile,
705 1434 2 2 ent_k_outfile,
706 1435 2 2 ent_k_number,
707 1436 2 2 ent_k_privilege,
708 1437 2 2 ent_k_datetime,
709 1438 2 2 ent_k_protection,
710 1439 2 2 ent_k_process,
711 1440 2 2 ent_k_inlog,
712 1441 2 2 ent_k_outlog,
713 1442 2 2 ent_k_insym,

```

```

: 714      1443 2      ent_k_outsym,
: 715      1444 2      ent_k_node,
: 716      1445 2      ent_k_device,
: 717      1446 2      ent_k_dir,
: 718      1447 2      ent_k_uic,
: 719      1448 2      ent_k_restofline,
: 720      1449 2      ent_k_parenvalue,
: 721      1450 2      ent_k_deltatime,
: 722      1451 2      ent_k_quotedstring,
: 723      1452 2      ent_k_file,
: 724      1453 2      ent_k_expression,
: 725      1454 2      ent_k_test1,
: 726      1455 2      ent_k_test2,
: 727      1456 2      ent_k_test3,
: 728      1457 2      ent_k_acl
: 729      1458 2      ));
: 730      1459 2
: 731      1460 2
: 732      1461 2      ! Sit in a loop searching the table of type names for the symbol in the
: 733      1462 2      ! token buffer.  If a match is found, return its corresponding number.
: 734      1463 2
: 735      1464 2      incru i from 1 to ent_k_max_valtype do (
: 736      1465 2          bind
: 737      1466 2              name = .type_names[.i]: vector[,byte];
: 738      1467 2
: 739      1468 2              if ch$eq1(.cdu$gq_token[.len],.cdu$gq_token[ptr],.name[0],name[1],') then
: 740      1469 2                  return .type_codes[.i];
: 741      1470 2      );
: 742      1471 2
: 743      1472 2      ! The user specified an invalid builtin type.
: 744      1473 2
: 745      1474 2      cdu$report_syntax_error(msg(cdu$_invtype),1,cdu$gq_token);
: 746      1475 2      return 0;
: 747      1476 2
: 748      1477 1      END;

```

.PSECT \$PLITS,NOWRT,NOEXE,2

										00	000B4	P.AAY:	.ASCII	<0>	
										07	000B5	P.AAZ:	.ASCII	<7>\\$INFILE\	
										08	000BD	P.ABA:	.ASCII	<8>\\$OUTFILE\	
										07	000C6	P.ABB:	.ASCII	<7>\\$NUMBER\	
										0A	000CE	P.ABC:	.ASCII	<10>\\$PRIVILEGE\	
										09	000D9	P.ABD:	.ASCII	<9>\\$DATETIME\	
										0B	000E3	P.ABE:	.ASCII	<11>\\$PROTECTION\	
										08	000EF	P.ABF:	.ASCII	<8>\\$PROCESS\	
										06	000F8	P.ABG:	.ASCII	<6>\\$INLOG\	
										07	000FF	P.ABH:	.ASCII	<7>\\$OUTLOG\	
										06	00107	P.ABI:	.ASCII	<6>\\$INSYM\	
										07	0010E	P.ABJ:	.ASCII	<7>\\$OUTSYM\	
										05	00116	P.ABK:	.ASCII	<5>\\$NODE\	
										07	0011C	P.ABL:	.ASCII	<7>\\$DEVICE\	
										0A	00124	P.ABM:	.ASCII	<10>\\$DIRECTORY\	
										04	0012F	P.ABN:	.ASCII	<4>\\$UIC\	
										0D	00134	P.ABO:	.ASCII	<13>\\$REST_OF_LINE\	

PARSE2
V04-000

N 11
15-Sep-1984 23:48:10
14-Sep-1984 11:58:26

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[CDU.SRC]PARSE2.B32;1

Page 28
(9)

```
44 45 5A 49 53 45 48 54 4E 45 52 41 50 24 14 00142 P.ABP: .ASCII <20>\$PARENTHESED_VALU\
45 55 4C 41 56 5F 00151
47 4E 49 52 54 53 5F 44 45 54 4F 55 51 24 0A 00157 P.ABQ: .ASCII <10>\$DELTA TIME\
54 53 5F 44 45 54 4F 55 51 24 0E 00162 P.ABR: .ASCII <14>\$QUOTED_STRING\
4E 4F 49 53 53 45 52 50 58 45 24 05 00171 P.ABS: .ASCII <5>\$FILE\
31 54 53 45 54 24 24 0B 00177 P.ABT: .ASCII <11>\$EXPRESSION\
32 54 53 45 54 24 24 07 00183 P.ABU: .ASCII <7>\$TEST1\
33 54 53 45 54 24 24 07 00188 P.ABV: .ASCII <7>\$TEST2\
4C 43 41 24 04 00193 P.ABW: .ASCII <7>\$TEST3\
00198 P.ABX: .ASCII <4>\$ACL\
```

.PSECT \$OWNS,NOEXE,2

00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00000 TYPE_NAMES:

```
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00018 .ADDRESS P.AAY, P.AAZ, P.ABA, P.ABB, P.ABC, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00030 P.ABD, P.ABE, P.ABF, P.ABG, P.ABH, P.ABI, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00048 P.ABJ, P.ABK, P.ABL, P.ABM, P.ABN, P.ABO, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00060 P.ABP, P.ABQ, P.ABR, P.ABS, P.ABT, P.ABU, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00068 P.ABV, P.ABW, P.ABX
```

OE OD OC OB OA 09 08 07 06 05 04 03 02 01 00 00068 TYPE_CODES:

```
19 18 17 16 15 14 13 12 11 10 0F 00077 .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
```

.EXTRN CDU\$_INVTYPE

.PSECT \$CODE\$,NOWRT,2

007C 00000 CDU\$BUILTIN TYPE:

```
56 00000000G 00 9E 00002 .WORD Save R2,R3,R4,R5,R6 1398
55 04 A6 D0 00009 MOVAB CDU$GQ_TOKEN, R6
54 01 D0 0000D MOVL CDU$GQ_TOKEN+4, R5 1468
50 0000'CF44 D0 00010 1$: MOVL #1, 1
51 60 9A 00016 MOVL TYPE_NAMES[1], R0 1466
65 66 2D 00019 MOVZBL (R0), R1 1468
01 A0 0001E CMPC5 CDU$GQ_TOKEN, (R5), #32, R1, 1(R0)
07 12 00020 BNEQ 2$
50 0000'CF44 9A 00022 MOVZBL TYPE_CODES[1], R0 1469
04 00028 RET
19 54 D6 00029 2$: INCL 1 1464
54 D1 0002B CMPL 1, #25
E0 1B 0002E BLEQU 1$
56 DD 00030 PUSHL R6 1474
01 DD 00032 PUSHL #1
00000000G 00 00000000G 8F DD 00034 PUSHL #CDU$_INVTYPE
03 FB 0003A CALLS #3, CDU$REPORT_SYNTAX_ERROR
50 D4 00041 CLRL R0 1475
04 00043 RET 1477
```

; Routine Size: 68 bytes. Routine Base: \$CODE\$ + 0708

```

750 1478 1 1++
751 1479 1 1 Description: This syntax routine recognizes the 'cli-flag' construct,
752 1480 1 1 which is simply one of the allowable CLI flags.
753 1481 1 1
754 1482 1 1 Parameters: parent By reference, parent node of this construct.
755 1483 1 1
756 1484 1 1 Returns: A node representing the flag.
757 1485 1 1
758 1486 1 1 Notes:
759 1487 1 1 --
760 1488 1 1
761 1489 1 1 GLOBAL ROUTINE cdu$cli_flag(parent: ref node)
762 1490 2 2 = BEGIN
763 1491 2 2
764 1492 2 2 local
765 1493 2 2 flag: ref node;
766 1494 2 2
767 1495 2 2
768 1496 2 2 ! Determine which CLI flag we have.
769 1497 2 2
770 1498 3 3 flag = (if token_is(tkn_k_symbol,'ABBREVIATE') then
771 1499 3 3 cdu$create_node(node_k_abbrev)
772 1500 3 3 else if token_is(tkn_k_symbol,'FOREIGN') then
773 1501 3 3 cdu$create_node(node_k_foreign)
774 1502 3 3 else if token_is(tkn_k_symbol,'IMMEDIATE') then
775 1503 3 3 cdu$create_node(node_k_immed)
776 1504 3 3 else if token_is(tkn_k_symbol,'MCRIGNORE') then
777 1505 3 3 cdu$create_node(node_k_mcrignore)
778 1506 3 3 else if token_is(tkn_k_symbol,'MCROPTDELIM') then
779 1507 3 3 cdu$create_node(node_k_mcroptdelim)
780 1508 3 3 else if token_is(tkn_k_symbol,'MCRPARSE') then
781 1509 3 3 cdu$create_node(node_k_mcrparse)
782 1510 3 3 else if token_is(tkn_k_symbol,'NOSTATUS') then
783 1511 3 3 cdu$create_node(node_k_nostat)
784 1512 3 3 else
785 1513 3 3 cdu$create_node(node_k_error)
786 1514 3 3 );
787 1515 2 2
788 1516 2 2 ! If we didn't find one, that's an error. Go on to the next token.
789 1517 2 2
790 1518 2 2 if .flag[node_w_type] eqlu node_k_error then
791 1519 2 2 cdu$report_syntax_error(msg(cdu$_invcliflag),1,cdu$gq_token);
792 1520 2 2 cdu$token_must_be(tkn_k_symbol);
793 1521 2 2 return .flag;
794 1522 2 2
795 1523 1 1 END;

```

.PSECT SPLITS,NOWRT,NOEXE,2

45	54	41	49	56	45	52	42	42	41	001A0	P.ABY:	.ASCII	\ABBREVIATE\	
			4E	47	49	45	52	4F	46	001AA	P.ABZ:	.ASCII	\FOREIGN\	
	45	54	41	49	44	45	4D	4D	49	001B1	P.ACA:	.ASCII	\IMMEDIATE\	
	45	52	4F	4E	47	49	52	43	4D	001BA	P.ACB:	.ASCII	\MCRIGNORE\	
4D	49	4C	45	44	54	50	4F	52	43	4D	001C3	P.ACC:	.ASCII	\MCROPTDELIM\
			45	53	52	41	50	52	43	4D	001CE	P.ACD:	.ASCII	\MCRPARSE\

53	55	54	41	54	53	4F	4E	001D6	P.ACE:	.ASCII	\NOSTATUS\	:
										.EXTRN	CDU\$ INVCLIFLAG	
										.PSECT	\$CODE\$,NOWRT,2	
										.ENTRY	CDUSCLI FLAG, Save R2,R3,R4,R5,R6,R7	1489
										MOVAB	P.ABY, R7	
										MOVAB	CDUSGQ_TOKEN, R6	
										CLRL	R4	1498
										CMPL	CDUSGL_TOKEN_CLASS, #13	
										BNEQ	1\$	
										INCL	R4	
										MOVL	CDUSGQ_TOKEN+4, R0	
										CMPC5	CDUSGQ_TOKEN, (R0), #0, #10, P.ABY	
										BNEQ	1\$	
										PUSHL	#34	1499
										BRB	8\$	
										BLBC	R4, 7\$	1500
										MOVL	CDUSGQ_TOKEN+4, R0	
										CMPC5	CDUSGQ_TOKEN, (R0), #0, #7, P.ABZ	
										BNEQ	2\$	
										PUSHL	#35	1501
										BRB	8\$	
										BLBC	R4, 7\$	1502
										MOVL	CDUSGQ_TOKEN+4, R0	
										CMPC5	CDUSGQ_TOKEN, (R0), #0, #9, P.ACA	
										BNEQ	3\$	
										PUSHL	#36	1503
										BRB	8\$	
										BLBC	R4, 7\$	1504
										MOVL	CDUSGQ_TOKEN+4, R0	
										CMPC5	CDUSGQ_TOKEN, (R0), #0, #9, P.ACB	
										BNEQ	4\$	
										PUSHL	#37	1505
										BRB	8\$	
										BLBC	R4, 7\$	1506
										MOVL	CDUSGQ_TOKEN+4, R0	
										CMPC5	CDUSGQ_TOKEN, (R0), #0, #11, P.ACC	
										BNEQ	5\$	
										PUSHL	#38	1507
										BRB	8\$	
										BLBC	R4, 7\$	1508
										MOVL	CDUSGQ_TOKEN+4, R0	
										CMPC5	CDUSGQ_TOKEN, (R0), #0, #8, P.ACD	
										BNEQ	6\$	
										PUSHL	#39	1509
										BRB	8\$	
										BLBC	R4, 7\$	1510
										MOVL	CDUSGQ_TOKEN+4, R0	
										CMPC5	CDUSGQ_TOKEN, (R0), #0, #8, P.ACE	

```

36 A7 0009B
04 12 0009D BNEQ 7$
28 DD 0009F PUSHL #40 1511
02 11 000A1 BRB 8$
7E D4 000A3 7$: CLRL -(SP) 1513
01 FB 000A5 8$: CALLS #1, CDUS$CREATE_NODE
50 D0 000AC MOVL R0, FLAG
65 B5 000AF TSTW (FLAG) 1518
11 12 000B1 BNEQ 9$
56 DD 000B3 PUSHL R6 1519
01 DD 000B5 PUSHL #1
0F DD 000B7 PUSHL #CDUS$ INVCLIFLAG
03 FB 000BD CALLS #3, CDUS$REPORT_SYNTAX_ERROR
0D DD 000C4 9$: PUSHL #13 1520
01 FB 000C6 CALLS #1, CDUS$TOKEN_MUST_BE
55 D0 000CD MOVL FLAG, R0 1521
04 000D0 RET 1523

```

; Routine Size: 209 bytes, Routine Base: \$CODE\$ + 074C

```

; 796 1524 1 END
; 797 1525 0 ELUDOM

```

PSECT SUMMARY

Name	Bytes	Attributes
\$SPLITS	478	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODE\$	2077	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$OWNS	130	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	Symbols		Pages Mapped	Processing Time
	Total	Loaded Percent		
_S255\$DUA28:[SYSLIB]LIB.L32;1	18619	4 0	1000	00:01.9

```

; Information: 2
; Warnings: 0
; Errors: 0

```

COMMAND QUALIFIERS

PARSE2
V04-000

E 12
15-Sep-1984 23:48:10
14-Sep-1984 11:58:26

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[CDU.SRC]PARSE2.B32;1 Page 32
(10)

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:PARSE2/OBJ=OBJ\$:PARSE2 MSRC\$:PARSE2/UPDATE=(ENH\$:PARSE2)

; Size: 2077 code + 608 data bytes
; Run Time: 00:37.1
; Elapsed Time: 01:21.0
; Lines/CPU Min: 2467
; Lexemes/CPU-Min: 22234
; Memory Used: 234 pages
; Compilation Complete

0044 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

