

[illegible]

```
DDDDDDDD  BBBB BBBB  GGGGGGGG  NN  NN  SSSSSSSS  EEEEEEEEE  TTTTTTTTTT
DDDDDDDD  BBBB BBBB  GGGGGGGG  NN  NN  SSSSSSSS  EEEEEEEEE  TTTTTTTTTT
DD  DD  BB  BB  GG  NN  NN  SS  EE  TT
DD  DD  BB  BB  GG  NN  NN  SS  EE  TT
DD  DD  BB  BB  GG  NNNN  NN  SS  EE  TT
DD  DD  BB  BB  GG  NNNN  NN  SS  EE  TT
DD  DD  BBBB BBBB  GG  NN  NN  SSSSSS  EEEEEEE  TT
DD  DD  BBBB BBBB  GG  NN  NN  SSSSSS  EEEEEEE  TT
DD  DD  BB  BB  GG  GGGGGG  NN  NNNN  SS  EE  TT
DD  DD  BB  BB  GG  GGGGGG  NN  NNNN  SS  EE  TT
DD  DD  BB  BB  GG  GG  NN  NN  SS  EE  TT
DD  DD  BB  BB  GG  GG  NN  NN  SS  EE  TT
DDDDDDDD  BBBB BBBB  GGGGGG  NN  NN  SSSSSSSS  EEEEEEEEE  TT
DDDDDDDD  BBBB BBBB  GGGGGG  NN  NN  SSSSSSSS  EEEEEEEEE  TT
```

```
....
....
....
....
```

```
LL  IIIII  SSSSSSSS
LL  IIIII  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
LLLLLLLLLL  IIIII  SSSSSSSS
LLLLLLLLLL  IIIII  SSSSSSSS
```



```
1 0001 0 MODULE DBGNSET (IDENT = 'V04-000') =
2 0002 1 BEGIN
3 0003 1
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: DEBUG
30 0030 1
31 0031 1 ABSTRACT:
32 0032 1
33 0033 1 This module contains the ATN parse network and the command execution network
34 0034 1 for the SET verb. These are currently the largest and most complicated
35 0035 1 networks. Briefly, the parse network constructs a command execution tree
36 0036 1 which consists of at least a verb node, and possibly one or more noun nodes
37 0037 1 The execution network accepts this tree as input and decodes the arguments
38 0038 1 of the various nodes to perform the specified command.
39 0039 1
40 0040 1 ENVIRONMENT: VAX/VMS
41 0041 1
42 0042 1 AUTHOR: David Plummer , CREATION DATE: 3/28/80
43 0043 1
44 0044 1 VERSION: V02.2-001
45 0045 1
46 0046 1 MODIFIED BY:
47 0047 1
48 0048 1 Richard Title 31-Jul-81
49 0049 1 Sid Maxwell 07-May-82
50 0050 1 Brad Becker 13-Sep-83
51 0051 1
52 0052 1 REVISION HISTORY:
53 0053 1 3.0 31-JUL-81 RT Added new nouns FLOAT, D_FLOAT, etc.
54 0054 1 in the SET TYPE command.
55 0055 1 3.1 13-SEP-81 RT Implemented new SET commands:
56 0056 1 SET STEP SOURCE
57 0057 1 SET SOURCE dir-list
```

```
58 0058 1 : SET MAX_SOURCE_FILES n
59 0059 1 : as part of source line display
60 0060 1 : 3.2 9-Oct-81 RT Implemented SET MARGINS
61 0061 1 : 3.3 13-Oct-81 RT Implemented SET SEARCH
62 0062 1 : 3B.0 18-Feb-82 PS Implemented SET DEVELOPER
63 0063 1 : ??? 06-May-82 RT Implemented SET DEFINE
64 0064 1 : 07-May-82 SRM Added hooks for new:
65 0065 1 : SET BREAK
66 0066 1 : SET TRACE
67 0067 1 : SET WATCH
68 0068 1 : 09-Aug-82 RT Added implementation level 3 code for
69 0069 1 : SET BREAK, SET TRACE, SET WATCH
70 0070 1 : 15-Apr-83 RT Added SET SCOPE/MODULE
71 0071 1 : 22-Apr-83 RT Added SET MODULE/ALLOCATE
72 0072 1 : 15-Jul-83 WC3 Added SET TYPE PACKED:n and SET TYPE/OVERRIDE PACKED:n
73 0073 1 : 18-Jul-83 WC3 Added SET TYPE DATE_TIME and SET TYPE/OVERRIDE DATE_TIME
74 0074 1 : 17-Aug-83 PS Added SET MODE G/NOG_FLOAT.
75 0075 1 : 4.0 13-Sep-83 BAB Added SET KEY.
76 0076 1 :
77 0077 1 :
78 0078 1 REQUIRE 'SRC$:DBGPROLOG.REQ';
79 0212 1 :
80 0213 1 LIBRARY 'LIB$:DBGGEN.L32';
81 0214 1 :
82 0215 1 FORWARD ROUTINE
83 0216 1 DBG$NPARSE_SET,
84 0217 1 DBG$NEXECUTE_SET,
85 0218 1 DBG$NSET_LOG,
86 0219 1 DBG$NSETOP_LOG,
87 0220 1 DUMMY: NOVALUE;
88 0221 1 :
: SET verb parse network
: SET verb execution network
: Sets up an output log
: Opens and connects an output log
: Dummy routine to foil BLISS
: compiler optimization
```


90	0222	1	EXTERNAL ROUTINE	
91	0223	1	DBG\$EVENT_SYNTAX,	Event syntax (parser)
92	0224	1	DBG\$EVENT_SEMANTICS,	Event semantics
93	0225	1	DBG\$NGET_LENGTH,	Returns bit length for a primary
94	0226	1	DBG\$NGET_DIR_LIST,	Parses a directory list for the
95	0227	1		SET SOURCE dir-list command
96	0228	1	DBG\$NGET_TRANS_RADIX,	Translate radix
97	0229	1	DBG\$SCR_EXECUTE_DISPLAY_CMD:NOVALUE,	Execute the SET DISPLAY command
98	0230	1	DBG\$SCR_EXECUTE_SETTERM_CMD:NOVALUE,	Execute the SET TERMINAL command
99	0231	1	DBG\$SCR_EXECUTE_SETWIND_CMD:NOVALUE,	Execute the SET WINDOW command
100	0232	1	DBG\$SCR_PARSE_DISPLAY_CMD:NOVALUE,	Parse the SET DISPLAY command
101	0233	1	DBG\$SCR_PARSE_SETTERM_CMD:NOVALUE,	Parse the SET TERMINAL command
102	0234	1	DBG\$SCR_PARSE_SETWIND_CMD:NOVALUE,	Parse the SET WINDOW command
103	0235	1	DBG\$SCR_SCREEN_MODE:NOVALUE,	Set or clear screen mode
104	0236	1	DBG\$STA_GETSOURCEMOD,	Finds a Module RST pointer
105	0237	1	DBG\$SRC_SET_SOURCE:NOVALUE,	Implements the SET SOURCE command
106	0238	1	DBG\$NSAVE_DECIMAL_INTEGER,	Saves an inline numeric literal
107	0239	1	DBG\$NSAVE_INTEGER,	Saves a numeric literal in the current radix
108	0240	1	DBG\$NSAVE_STRING,	Saves a counted string from input
109	0241	1	DBG\$NGET_TYPE,	Obtains type of primary
110	0242	1	DBG\$PRINT:NOVALUE,	Format print output
111	0243	1	DBG\$NEWLINE:NOVALUE,	Output a line of print output
112	0244	1	DBG\$SET_SEARCH_LVL:NOVALUE,	Sets search data structure pointer
113	0245	1	DBG\$SET_STP_LVL:NOVALUE,	Sets step data structure pointer
114	0246	1	DBG\$SET_DEFINE_LVL:NOVALUE,	Sets define data structure pointer
115	0247	1	DBG\$SET_MOD_LVL,	Sets mode structure level
116	0248	1	DBG\$NPARSE_SET_TASK:NOVALUE,	Parses SET TASK command
117	0249	1	DBG\$NEXECUTE_SET_TASK:NOVALUE,	Handles SET TASK command
118	0250	1	DBG\$RST_SETSCOPE:NOVALUE,	Sets user specified scopes
119	0251	1	DBG\$NPARSE_SCOPE_LIST,	Parses a scope list
120	0252	1	DBG\$NSHOW_MARGINS:NOVALUE,	Shows the margins
121	0253	1	DBG\$SRC_SET_MAX_FILES:NOVALUE,	Performs SET MAX command
122	0254	1	DBG\$NPATHDESC_TO_CS:NOVALUE,	Translates a pathname desc to a c.s.
123	0255	1	DBG\$RST_SETMOD,	Sets a module
124	0256	1	DBG\$GET_MEMORY,	Allocates a permanent memory block
125	0257	1	DBG\$REL_MEMORY:NOVALUE,	Releases a memory block
126	0258	1	DBG\$GET_TEMPMEM,	Allocates and lists dynamic storage
127	0259	1	DBG\$NSAVE_FILESP,	Eats and stores a filespec string
128	0260	1	DBG\$NMATCH,	Routine to match keywords
129	0261	1	DBG\$NPARSE_ADDRESS,	Interface routine to Address Expression Interpreter
130	0262	1	DBG\$NNEXT_WORD,	Isolates next word of input for syntax errors
131	0263	1	DBG\$NOUT_INFO,	Outputs an informational message
132	0264	1	DBG\$NMAKE_ARG_VECT,	Constructs a message argument vector
133	0265	1	DBG\$NSYNTAX_ERROR,	Constructs a syntax error message
134	0266	1	DBG\$SET_LANG,	Version 2 routine to change language
135	0267	1	DBG\$READ_KEY_INFO,	Reads in the state-name string for SET KEY
136	0268	1	SMG\$SET_DEFAULT_STATE,	Sets the default keypad state
137	0269	1	SMG\$SET_KEYPAD_MODE;	Sets terminal to numeric/applicat. mode
138	0270	1		
139	0271	1	EXTERNAL	
140	0272	1	DBG\$GB_KEYPAD_INPUT: BYTE,	TRUE if keypad input is enabled
141	0273	1	DBG\$GL_KEY_TABLE_ID,	Used in SET KEY
142	0274	1	DBG\$GL_KEYBOARD_ID,	Used in SET MODE KEY
143	0275	1	DBG\$GB_RADIX: VECTOR[3, BYTE],	Override and default radix
144	0276	1	DBG\$GL_GBLTYP,	Override type
145	0277	1	DBG\$GW_GBLLENTH: WORD,	Override length
146	0278	1	DBG\$GL_DFLTYP,	Default type


```
147 0279 1 DBG$GW_DFLTLENG: WORD, ! Default length
148 0280 1 DBG$GB_LANGUAGE: BYTE, ! Current language
149 0281 1 DBG$GB_SEARCH_PTR: REF VECTOR[,BYTE], ! Search structure
150 0282 1 DBG$GB_MOD_PTR: REF VECTOR[,BYTE], ! Mode structure
151 0283 1 DBG$GB_DEFINE_PTR: REF VECTOR[,BYTE], ! Define structure
152 0284 1 DBG$RUNFRAME: BLOCK[,BYTE], ! Current runframe
153 0285 1 DBG$GB_RESIGNAL: BYTE, ! If TRUE, then resignal exeptions
154 0286 1 DBG$GL_LOG_BUF, ! Old debugger pointer to counted string filespec
155 0287 1 DBG$GL_CONTEXT: BITVECTOR, ! Old debugger context switches
156 0288 1 DBG$GB_DEF_OUT: VECTOR[,BYTE], ! Old debugger output control vector
157 0289 1 DBG$GL_LOGFAB: BLOCK[,BYTE], ! FAB for log file
158 0290 1 DBG$GL_LOGRAB: BLOCK[,BYTE], ! RAB for log file
159 0291 1 DBG$GL_LOGNAM: REF $NAM DECL, ! NAM block for file spec
160 0292 1 DBG$GB_LOGFSR: REF VECTOR[,BYTE], ! Resultant LOG filespec buffer
161 0293 1 DBG$GB_LOGFSE: REF VECTOR[,BYTE], ! Expanded LOG filespec buffer
162 0294 1 DBG$GL_SCREEN_LOG, ! Flag set to activate screen logging
163 0295 1 DBG$GB_STP_PTR: ! Pointer to current stepping descriptor
164 0296 1 REF EVENT$STEPPING_DESCRIPTOR,
165 0297 1 DBG$SRC_LEFT_MARGIN, ! Left margin for source display
166 0298 1 DBG$SRC_RIGHT_MARGIN, ! Right margin for source display
167 0299 1 DBG$GL_ORIG_COMMAND_PTR, ! Pointer to original command string
168 0300 1 DBG$GL_UPCASE_COMMAND_PTR: VECTOR[2],
169 0301 1 ! Pointers to start and end
170 0302 1 ! of current command string
171 0303 1
172 0304 1
173 0305 1
174 0306 1
175 0307 1 ! These are composite verb literals used in parsing and execution.
176 0308 1
177 0309 1 ! Note - you may cause yourself problems if you try to renumber these,
178 0310 1 ! because the EVENT$K_SET_XXX definitions in DBGLIB.REQ must correspond
179 0311 1 ! to the numbers here.
180 0312 1
181 0313 1 LITERAL
182 0314 1 SET_MINIMUM = 1,
183 0315 1 SET_BREAK = 1, ! Also defined as EVENT$K_SET_BREAK
184 0316 1 SET_BREAK_DO = 2,
185 0317 1 SET_EXCEPTION_BREAK = 3, ! Also defined as EVENT$K_SET_BREAK_EXC
186 0318 1 SET_LANGUAGE = 4,
187 0319 1 SET_LOG = 5,
188 0320 1 SET_MODE = 6,
189 0321 1 SET_MODULE = 7,
190 0322 1 SET_MODULE_ALL = 8,
191 0323 1 SET_OUTPUT = 9,
192 0324 1 SET_RADIX = 27,
193 0325 1 SET_RADIX_OVERRIDE = 28,
194 0326 1 SET_SCOPE = 10,
195 0327 1 SET_STEP = 11,
196 0328 1 SET_TYPE = 12,
197 0329 1 SET_TYPE_OVERRIDE = 13,
198 0330 1 SET_TRACE = 14, ! Also defined as EVENT$K_SET_TRACE
199 0331 1 SET_TRACE_CALLS = 15,
200 0332 1 SET_TRACE_BRANCH = 16,
201 0333 1 SET_WATCH = 17, ! Also defined as EVENT$K_SET_WATCH
202 0334 1 SET_SOURCE = 18,
203 0335 1 SET_MAX_SOURCE_FILES = 19,
```



```
204 0336 1 SET_MARGINS = 20;
205 0337 1 SET_SEARCH = 21;
206 0338 1 SET_DEVELOPER = 22;
207 0339 1 SET_DEFINE = 23;
208 0340 1 SET_DISPLAY = 24;
209 0341 1 SET_TERMINAL = 25;
210 0342 1 SET_WINDOW = 26;
211 0343 1 SET_KEY = 29;
212 0344 1 SET_TASK = 30;
213 0345 1 SET_MAXIMUM = 30;
214 0346 1
215 0347 1 ! These are the legal mode literals for the SET MODE command.
216 0348 1 !
217 0349 1 LITERAL
218 0350 1 MODE_LOWEST = 1;
219 0351 1 MODE_BINARY = 1;
220 0352 1 MODE_OCTAL = 2;
221 0353 1 MODE_DECIMAL = 3;
222 0354 1 MODE_HEX = 4;
223 0355 1 MODE_SYMBOLS = 5;
224 0356 1 MODE_NOSYMBOLS = 6;
225 0357 1 MODE_DEFAULT = 7;
226 0358 1 MODE_SCREEN = 8;
227 0359 1 MODE_NOSCREEN = 9;
228 0360 1 MODE_KEYPAD = 10;
229 0361 1 MODE_NOKEYPAD = 11;
230 0362 1 MODE_G_FLOAT = 12;
231 0363 1 MODE_NOG_FLOAT = 13;
232 0364 1 MODE_HIGHEST = 13;
233 0365 1
234 0366 1 GLOBAL
235 0367 1 DBG$GL_DEVELOPER: ! Special mode bits for developers
236 0368 1 BITVECTOR[32] INITIAL(0); !
237 0369 1
238 0370 1 MACRO
239 0371 1
240 0372 1 ! OUTPUT_RMS_ERROR reports RMS errors caused by $CREATE, $OPEN, and $CONNECT
241 0373 1 !
242 M 0374 1 OUTPUT_RMS_ERROR (BLOCK_TYPE) =
243 M 0375 1 BEGIN
244 M 0376 1 LOCAL
245 M 0377 1 MSG_DESC : REF dbg$stg_desc,
246 M 0378 1 STS;
247 M 0379 1 STV;
248 M 0380 1
249 M 0381 1 msg_desc = dbg$get_tempmem (2);
250 M 0382 1
251 M 0383 1 build_error_desc (msg_desc);
252 M 0384 1 IF block_type EQL 1
253 M 0385 1 THEN
254 M 0386 1 BEGIN
255 M 0387 1 sts = .dbg$gl_logfab [fab$l_sts];
256 M 0388 1 stv = .dbg$gl_logfab [fab$l_stv];
257 M 0389 1 END
258 M 0390 1 ELSE
259 M 0391 1 BEGIN
260 M 0392 1 sts = .dbg$gl_lograb [rab$l_sts];
```

```
261      M 0393 1      stv = .dbg$gl_lograb [rab$l_stv];
262      M 0394 1      END;
263      M 0395 1
264      M 0396 1      .message_vect = dbg$nmake_arg_vect (shr$openout+dbg_fac_code,
265      M 0397 1      1, .msg_desc, .sts, .stv);
266      M 0398 1
267      M 0399 1      RETURN sts$k_severe;
268      M 0400 1
269      M 0401 1      END %,
270      M 0402 1
271      M 0403 1      ! Build_error_desc creates a descriptor for RMS error reporting
272      M 0404 1
273      M 0405 1      BUILD_ERROR_DESC (MSG_DESC) =
274      M 0406 1      IF .dbg$gl_log_buf NEQ 0
275      M 0407 1      THEN
276      M 0408 1      BEGIN
277      M 0409 1      msg_desc [dsc$w_length] = .dbg$gl_logfab [fab$b_fns];
278      M 0410 1      msg_desc [dsc$a_pointer] = .dbg$gl_logfab [fab$l_fna];
279      M 0411 1      END
280      M 0412 1      ELSE
281      M 0413 1      BEGIN
282      M 0414 1      msg_desc [dsc$w_length] = .dbg$gl_logfab [fab$b_dns];
283      M 0415 1      msg_desc [dsc$a_pointer] = .dbg$gl_logfab [fab$l_dna];
284      M 0416 1      END %,
285      M 0417 1
286      M 0418 1      ! Restore_nam restores an RMS NAM block
287      M 0419 1
288      M 0420 1      RESTORE_NAM =
289      M 0421 1
290      M 0422 1      BEGIN
291      M 0423 1
292      M 0424 1      LOCAL
293      M 0425 1      ERR,
294      M 0426 1      TYPE_B;
295      M 0427 1
296      M 0428 1      ! Release storage for the aborted log file
297      M 0429 1
298      M 0430 1      dbg$rel_memory (.temp_nam);
299      M 0431 1      dbg$rel_memory (.temp_fsr);
300      M 0432 1      dbg$rel_memory (.temp_fse);
301      M 0433 1
302      M 0434 1      dbg$gl_lognam = .old_nam_ptr;
303      M 0435 1      old_nam_ptr = 0;
304      M 0436 1      dbg$gl_logfab [fab$l_nam] = .dbg$gl_lognam;
305      M 0437 1      dbg$gl_logfab [fab$l_fna] = .fna;
306      M 0438 1      dbg$gl_logfab [fab$b_fns] = .fns;
307      M 0439 1
308      M 0440 1      dbg$gl_logfab [fab$v_nam] = 1;
309      M 0441 1      dbg$gl_logfab [fab$v_cif] = 1;
310      M 0442 1      dbg$gl_logfab [fab$v_mxv] = 0;
311      M 0443 1
312      M 0444 1      err = dbg$nsetup_log (type_b);
313      M 0445 1
314      M 0446 1      IF NOT .err
315      M 0447 1      THEN output_rms_error (.type_b);
316      M 0448 1
317      M 0449 1      dbg$gb_def_out [out_log] = .log_temp;
```


DBGNSET
V04-000

K 14
16-Sep-1984 01:58:19
14-Sep-1984 12:17:21

VAX-11 Bliss-32 V4.0-742
[DEBUG.SRC]DBGNSET.B32;1

Page 7
(2)

: 318
: 319

M 0450 1
0451 1

END %;

```
321 0452 1 GLOBAL ROUTINE DBG$NPARSE_SET (INPUT_DESC, VERB_NODE, MESSAGE_VECT) =
322 0453 1
323 0454 1 ++
324 0455 1 FUNCTIONAL DESCRIPTION:
325 0456 1
326 0457 1 This routine comprises the ATN parse network for the SET verb. The network
327 0458 1 recognizes various keywords and constructs appropriate noun nodes which it
328 0459 1 links to the verb node. Many composite verbs are recognized in this network
329 0460 1 as well. The value of these composite verbs are placed in the verb node (verb composite)
330 0461 1 field.
331 0462 1
332 0463 1 FORMAL PARAMETERS:
333 0464 1
334 0465 1 INPUT_DESC - The present command line input descriptor
335 0466 1
336 0467 1 VERB_NODE - The first node in the command execution tree.
337 0468 1 Value field has already been filled in by
338 0469 1 DBG$NPARSE_CMD.
339 0470 1
340 0471 1 MESSAGE_VECT - The address of a longword to contain the address
341 0472 1 of a message argument vector
342 0473 1
343 0474 1 IMPLICIT INPUTS:
344 0475 1
345 0476 1 NONE
346 0477 1
347 0478 1 IMPLICIT OUTPUTS:
348 0479 1
349 0480 1 The command execution tree corresponding to the input command is constructed.
350 0481 1
351 0482 1 ROUTINE VALUE:
352 0483 1
353 0484 1 An unsigned integer longword completion code
354 0485 1
355 0486 1 COMPLETION CODES:
356 0487 1
357 0488 1 ST$K_SEVERE (4) - Unsuccessful parse
358 0489 1
359 0490 1 ST$K_SUCCESS (1) - Successful parse
360 0491 1
361 0492 1 SIDE EFFECTS:
362 0493 1
363 0494 1 NONE
364 0495 1
365 0496 1 --
366 0497 1
367 0498 2 BEGIN
368 0499 2
369 0500 2 ! Define strings used at this level.
370 0501 2
371 0502 2 BIND
372 0503 2 DBG$CS_BREAK = UPLIT BYTE (XASCIC 'BREAK'),
373 0504 2 DBG$CS_DEFINE = UPLIT BYTE (XASCIC 'DEFINE'),
374 0505 2 DBG$CS_DEVELOPER = UPLIT BYTE (XASCIC 'DEVELOPER'),
375 0506 2 DBG$CS_DISPLAY = UPLIT BYTE (XASCIC 'DISPLAY'),
376 0507 2 DBG$CS_EXCEPTION = UPLIT BYTE (XASCIC 'EXCEPTION'),
377 0508 2 DBG$CS_LANGUAGE = UPLIT BYTE (XASCIC 'LANGUAGE'),
```



```

378      0509      2      DBG$CS_LOG      = UPLIT BYTE (%ASCIC 'LOG'),
379      0510      2      DBG$CS_KEY      = UPLIT BYTE (%ASCIC 'KEY'),
380      0511      2      DBG$CS_MARGINS      = UPLIT BYTE (%ASCIC 'MARGINS'),
381      0512      2      DBG$CS_MAX_SOURCE_FILES = UPLIT BYTE (%ASCIC 'MAX_SOURCE_FILES'),
382      0513      2      DBG$CS_MODE      = UPLIT BYTE (%ASCIC 'MODE'),
383      0514      2      DBG$CS_MODULE      = UPLIT BYTE (%ASCIC 'MODULE'),
384      0515      2      DBG$CS_OUTPUT      = UPLIT BYTE (%ASCIC 'OUTPUT'),
385      0516      2      DBG$CS_RADIX      = UPLIT BYTE (%ASCIC 'RADIX'),
386      0517      2      DBG$CS_SCOPE      = UPLIT BYTE (%ASCIC 'SCOPE'),
387      0518      2      DBG$CS_SEARCH      = UPLIT BYTE (%ASCIC 'SEARCH'),
388      0519      2      DBG$CS_SOURCE      = UPLIT BYTE (%ASCIC 'SOURCE'),
389      0520      2      DBG$CS_STEP      = UPLIT BYTE (%ASCIC 'STEP'),
390      0521      2      DBG$CS_TASK      = UPLIT BYTE (%ASCIC 'TASK'),
391      0522      2      DBG$CS_TERMINAL      = UPLIT BYTE (%ASCIC 'TERMINAL'),
392      0523      2      DBG$CS_TRACE      = UPLIT BYTE (%ASCIC 'TRACE'),
393      0524      2      DBG$CS_TYPE      = UPLIT BYTE (%ASCIC 'TYPE'),
394      0525      2      DBG$CS_WATCH      = UPLIT BYTE (%ASCIC 'WATCH'),
395      0526      2      DBG$CS_WINDOW      = UPLIT BYTE (%ASCIC 'WINDOW'),
396      0527      2      DBG$CS_COLON      = UPLIT BYTE (%ASCIC ':'),
397      0528      2      DBG$CS_COMMA      = UPLIT BYTE (%ASCIC ','),
398      0529      2      DBG$CS_EQUAL      = UPLIT BYTE (%ASCIC '='),
399      0530      2      DBG$CS_SLASH      = UPLIT BYTE (%ASCIC '/'),
400      0531      2      DBG$CS_CR      = UPLIT BYTE (1, DBG$K_CAR_RETURN);
401      0532
402      0533      MAP
403      0534      VERB_NODE: REF DBG$VERB_NODE;      ! Pointer to command Verb Node
404      0535
405      0536      LOCAL
406      0537      NOUN_NODE: REF DBG$NOUN_NODE;      ! Pointer to current Noun Node
407      0538
408      0539
409      0540
410      0541      ! Construct the noun node.
411      0542
412      0543      NOUN_NODE = DBG$GET_TEMPMEM (DBG$K_NOUN_NODE_SIZE);
413      0544
414      0545
415      0546      ! Link the noun node to the verb node.
416      0547
417      0548      VERB_NODE [DBG$L_VERB_OBJECT_PTR] = .NOUN_NODE;
418      0549
419      0550
420      0551      ! Recognize keyword and transfer control to subnetwork if appropriate.
421      0552
422      0553      SELECTONE TRUE OF
423      0554      SET
424      0555
425      0556
426      0557      ! Parse the SET BREAK command.
427      0558
428      0559      [DBG$NMATCH (.INPUT_DESC, DBG$CS_BREAK, 1)]:
429      0560      BEGIN
430      0561      VERB_NODE [DBG$B_VERB_COMPOSITE] = EVENT$K_SET_BREAK;
431      0562      RETURN DBG$EVENT_SYNTAX (.INPUT_DESC,
432      0563      .VERB_NODE,
433      0564      .MESSAGE_VECT
434      0565      );

```



```

435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491

```

```

END;

[dbg$nmach (.input_desc, dbg$cs_define, 1)] :      ! Set Define
BEGIN
    ! Define the keywords that are legal after SET DEFINE
    !
    BIND
        dbg$cs_address = UPLIT BYTE (7, 'ADDRESS'),
        dbg$cs_command = UPLIT BYTE (7, 'COMMAND'),
        dbg$cs_procedure = UPLIT BYTE (9, 'PROCEDURE'),
        dbg$cs_string = UPLIT BYTE (6, 'STRING'),
        dbg$cs_value = UPLIT BYTE (5, 'VALUE');

    verb_node [dbg$b_verb_composite] = set_define;

    ! Pick up one keyword.
    ! *** Two of the five are commented out because they are
    ! *** not yet supported.
    !
    SELECTONE TRUE OF
        SET
            [dbg$nmach (.input_desc, dbg$cs_address, 1)]:
                noun_node [dbg$_noun_value] = define_address;

            [dbg$nmach (.input_desc, dbg$cs_command, 1)]:
                noun_node [dbg$_noun_value] = define_command;

            ! [dbg$nmach (.input_desc, dbg$cs_procedure, 1)]:
            !     noun_node [dbg$_noun_value] = define_procedure;

            ! [dbg$nmach (.input_desc, dbg$cs_string, 1)]:
            !     noun_node [dbg$_noun_value] = define_string;

            [dbg$nmach (.input_desc, dbg$cs_value, 1)]:
                noun_node [dbg$_noun_value] = define_value;

        [OTHERWISE]:
            BEGIN
                IF dbg$nmach (.input_desc, dbg$cs_cr, 1)
                THEN
                    .message_vect = dbg$nmach_arg_vect (dbg$_needmore)
                ELSE
                    .message_vect =
                        dbg$nsyntax_error (
                            dbg$nnext_word (.input_desc));
                RETURN sts$severe;
            END;

    TES;
END;

[dbg$nmach (.input_desc, dbg$cs_developer, 9)] :      ! Set Developer
BEGIN
    LOCAL

```



```
link;
verb_node [dbg$b_verb_composite] = set_developer;
link = verb_node[dbg$l_verb_object_ptr];
IF NOT dbg$nmatch(.input_desc, dbg$cs_cr, 1)
THEN
  BEGIN
    WHILE true DO
      BEGIN
        IF NOT dbg$nsave_integer(.input_desc, noun_node[dbg$l_noun_value])
        THEN
          RETURN sts$k_severe;

        IF (.noun_node[dbg$l_noun_value] LSS 0) OR
          (.noun_node[dbg$l_noun_value] GTR 31)
        THEN
          BEGIN
            .message_vect = dbg$nmake_arg_vect(dbg$_bitrange);
            RETURN sts$k_severe;
          END;

        link = noun_node[dbg$l_noun_link];
        IF NOT dbg$nmatch(.input_desc, dbg$cs_comma, 1)
        THEN
          BEGIN
            IF NOT dbg$nmatch(.input_desc, dbg$cs_cr, 1)
            THEN
              BEGIN
                .message_vect = dbg$nsyntax_error(dbg$nnext_word(.input_desc));
                RETURN sts$k_severe;
              END;

            ELSE
              EXITLOOP;

            END;

        noun_node = dbg$get_tempmem (dbg$k_noun_node_size);
        .link = .noun_node;
        END;
        ! End of WHILE loop.
      END
      ! End of SET DEVELOPER 0,1...
    ELSE
      BEGIN
        .message_vect = dbg$nmake_arg_vect(dbg$_needmore);
        RETURN sts$k_severe;
      END;

    .link = 0;
  END;

! Parse the SET DISPLAY command.
[DBG$NMATCH(.INPUT_DESC, DBG$CS_DISPLAY, 3)]:
```



```

549      BEGIN
550      VERB_NODE[DBG$B_VERB_COMPOSITE] = SET_DISPLAY;
551      DBG$SCR_PARSE_DISPLAY_CMD(.INPUT_DESC, TRUE, .VERB_NODE);
552      END;
553
554      ! Parse the SET EXCEPTION BREAK command.
555      [DBG$NMATCH (.INPUT_DESC, DBG$CS_EXCEPTION, 1)]:
556      BEGIN
557      ! We are looking for SET EXCEPTION BREAK
558      IF NOT dbg$nmatch (.input_desc, dbg$cs_break, 1)
559      THEN
560      BEGIN
561      !message_vect =
562      ( IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
563      THEN
564      dbg$nmake_arg_vect (dbg$_needmore)
565      ELSE
566      dbg$nsyntax_error (dbg$nnext_word (.input_desc));
567      RETURN sts$k_severe;
568      END;
569
570      ! Set the verb composite
571      verb_node [dbg$b_verb_composite] = set_exception_break;
572
573      ! Reset the noun and adverb pointers.
574      verb_node [dbg$l_verb_object_ptr] = 0;
575      verb_node [dbg$l_verb_adverb_ptr] = 0;
576      END;
577
578      ! Parse SET KEY
579      [dbg$nmatch (.input_desc, dbg$cs_key, 1)]:
580      BEGIN
581      MAP
582      input_desc          : REF BLOCK [,BYTE];
583
584      BIND
585      dbg$cs_state        = UPLIT BYTE (5, 'STATE'),
586      dbg$cs_nostate      = UPLIT BYTE (7, 'NOSTATE'),
587      dbg$cs_nolog        = UPLIT BYTE (5, 'NOLOG');
588
589      LOCAL
590      temp_key_desc      : REF dbg$stg_desc,
591      status;
592
593      IF NOT .dbg$gb_keypad_input
594      THEN
```



```

: 606
: 607
: 608
: 609
: 610
: 611
: 612
: 613
: 614
: 615
: 616
: 617
: 618
: 619
: 620
: 621
: 622
: 623
: 624
: 625
: 626
: 627
: 628
: 629
: 630
: 631
: 632
: 633
: 634
: 635
: 636
: 637
: 638
: 639
: 640
: 641
: 642
: 643
: 644
: 645
: 646
: 647
: 648
: 649
: 650
: 651
: 652
: 653
: 654
: 655
: 656
: 657
: 658
: 659
: 660
: 661
: 662

```

```

0737
0738
0739
0740
0741
0742
0743
0744
0745
0746
0747
0748
0749
0750
0751
0752
0753
0754
0755
0756
0757
0758
0759
0760
0761
0762
0763
0764
0765
0766
0767
0768
0769
0770
0771
0772
0773
0774
0775
0776
0777
0778
0779
0780
0781
0782
0783
0784
0785
0786
0787
0788
0789
0790
0791
0792
0793

```

```

        SIGNAL(dbg$_nokeydef);

verb_node [dbg$b_verb_composite] = set_key;

! Initialize the noun node to point to a descriptor of the
! current state.
temp_key_desc = dbg$get_tempmem(2);
temp_key_desc [dsc$w_length] = 0;
temp_key_desc [dsc$b_dtype] = dsc$k_dtype_t;
temp_key_desc [dsc$b_class] = dsc$k_class_d;
temp_key_desc [dsc$a_pointer] = 0;

smg$set_default_state (dbg$gl_key_table_id, 0, .temp_key_desc);

! Let this field indicate whether to output a log of the SET KEY command
verb_node [dbg$l_verb_adverb_ptr] = 1;

noun_node [dbg$l_noun_value] = .temp_key_desc;

WHILE (NOT dbg$match (.input_desc, dbg$cs_cr, 1)) AND
      (.input_desc [dsc$w_length] GTR 0) DO
    IF dbg$match (.input_desc, dbg$cs_slash, 1)
    THEN
        SELECTONE TRUE OF
        SET
            [dbg$match(.input_desc, dbg$cs_nolog, 3)] :
            BEGIN
                verb_node [dbg$l_verb_adverb_ptr] = 0;
            END;

            [dbg$match(.input_desc, dbg$cs_nostate, 3)] :
            BEGIN
                ! In this case the noun node field that points to
                ! the state name is reset anyway, as above.
                smg$set_default_state(dbg$gl_key_table_id, 0, .temp_key_desc);
                noun_node [dbg$l_noun_value] = .temp_key_desc;
            END;

            [dbg$match(.input_desc, dbg$cs_log, 1)] :
            BEGIN
                verb_node [dbg$l_verb_adverb_ptr] = 1;
            END;

            [dbg$match(.input_desc, dbg$cs_state, 1)] :
            BEGIN
                ! Check for =
                IF NOT dbg$match(.input_desc, dbg$cs_equal, 1)
                THEN
                    BEGIN
                        IF dbg$match(.input_desc, dbg$cs_cr, 1)
                        THEN

```



```
663 0794 5
664 0795 5
665 0796 5
666 0797 5
667 0798 4
668 0799 4
669 0800 4
670 0801 4
671 0802 4
672 0803 4
673 0804 4
674 0805 4
675 0806 4
676 0807 4
677 0808 3
678 0809 3
679 0810 3
680 0811 4
681 0812 4
682 0813 4
683 0814 4
684 0815 4
685 0816 4
686 0817 4
687 0818 3
688 0819 3
689 0820 3
690 0821 4
691 0822 4
692 0823 4
693 0824 3
694 0825 2
695 0826 2
696 0827 2
697 0828 2
698 0829 2
699 0830 2
700 0831 2
701 0832 2
702 0833 2
703 0834 2
704 0835 2
705 0836 2
706 0837 2
707 0838 2
708 0839 2
709 0840 2
710 0841 2
711 0842 2
712 0843 2
713 0844 2
714 0845 2
715 0846 2
716 0847 2
717 0848 2
718 0849 2
719 0850 3
```

```
ELSE .message_vect = dbg$make_arg_vect (dbg$_needmore)
ELSE .message_vect = dbg$syntax_error (dbg$next_word(.input_desc));
RETURN sts$k_severe;
END;

temp_key_desc [dsc$a_pointer] = 0;
temp_key_desc [dsc$w_length] = 0;
status = dbg$read_key_info (.input_desc,
                             .temp_key_desc,
                             .message_vect);

IF NOT .status
THEN
RETURN .status;
END;

[OTHERWISE] :
BEGIN
IF dbg$match(.input_desc, dbg$cs_cr, 1)
THEN
.message_vect = dbg$make_arg_vect (dbg$_needmore)
ELSE
.message_vect = dbg$syntax_error (dbg$next_word(.input_desc));
RETURN sts$k_severe;
END;

TES
ELSE
BEGIN
.message_vect = dbg$syntax_error (dbg$next_word(.input_desc));
RETURN sts$k_severe;
END;

END; ! End of SET KEY parse

[dbg$match (.input_desc, dbg$cs_language, 2)]:
BEGIN
BIND
DBG$CS_MACRO = UPLIT BYTE(%ASCIC 'MACRO'),
DBG$CS_FORTRAN = UPLIT BYTE(%ASCIC 'FORTRAN'),
DBG$CS_BLISS = UPLIT BYTE(%ASCIC 'BLISS'),
DBG$CS_COBOL = UPLIT BYTE(%ASCIC 'COBOL'),
DBG$CS_BASIC = UPLIT BYTE(%ASCIC 'BASIC'),
DBG$CS_PLI = UPLIT BYTE(%ASCIC 'PLI'),
DBG$CS_PASCAL = UPLIT BYTE(%ASCIC 'PASCAL'),
DBG$CS_C = UPLIT BYTE(%ASCIC 'C'),
DBG$CS_RPG = UPLIT BYTE(%ASCIC 'RPG'),
DBG$CS_ADA = UPLIT BYTE(%ASCIC 'ADA'),
DBG$CS_UNKNOWN = UPLIT BYTE(%ASCIC 'UNKNOWN');

VERB_NODE [DBG$B_VERB_COMPOSITE] = SET_LANGUAGE;

SELECTONE TRUE OF
SET
[DBG$NMATCH (.INPUT_DESC, DBG$CS_MACRO, 2)] :
```



```

: 720      0851      3
: 721      0852
: 722      0853
: 723      0854
: 724      0855
: 725      0856
: 726      0857
: 727      0858
: 728      0859
: 729      0860
: 730      0861
: 731      0862
: 732      0863
: 733      0864
: 734      0865
: 735      0866
: 736      0867
: 737      0868
: 738      0869
: 739      0870
: 740      0871
: 741      0872
: 742      0873
: 743      0874
: 744      0875
: 745      0876
: 746      0877
: 747      0878
: 748      0879
: 749      0880
: 750      0881
: 751      0882
: 752      0883
: 753      0884      3
: 754      0885      4
: 755      0886      4
: 756      0887      4
: 757      0888      4
: 758      0889      4
: 759      0890      4
: 760      0891      4
: 761      0892      4
: 762      0893      4
: 763      0894      4
: 764      0895      3
: 765      0896
: 766      0897
: 767      0898
: 768      0899
: 769      0900
: 770      0901
: 771      0902
: 772      0903
: 773      0904
: 774      0905
: 775      0906
: 776      0907      3

      NOUN_NODE [DBG$ _NOUN_VALUE] = DBG$K_MACRO;
[DBG$NMATCH (.INPUT_DESC, DBG$CS_FORTAN, 2)] :
      NOUN_NODE [DBG$ _NOUN_VALUE] = DBG$K_FORTAN;
[DBG$NMATCH (.INPUT_DESC, DBG$CS_BLISS, 2)] :
      NOUN_NODE [DBG$ _NOUN_VALUE] = DBG$K_BLISS;
[DBG$NMATCH (.INPUT_DESC, DBG$CS_COBOL, 2)] :
      NOUN_NODE [DBG$ _NOUN_VALUE] = DBG$K_COBOL;
[DBG$NMATCH (.INPUT_DESC, DBG$CS_BASIC, 2)] :
      NOUN_NODE [DBG$ _NOUN_VALUE] = DBG$K_BASIC;
[DBG$NMATCH (.INPUT_DESC, DBG$CS_PLI, 2)] :
      NOUN_NODE [DBG$ _NOUN_VALUE] = DBG$K_PLI;
[DBG$NMATCH (.INPUT_DESC, DBG$CS_PASCAL, 2)] :
      NOUN_NODE [DBG$ _NOUN_VALUE] = DBG$K_PASCAL;
[DBG$NMATCH(.INPUT_DESC, DBG$CS_C, 1)]:
      NOUN_NODE[DBG$ _NOUN_VALUE] = DBG$K_C;
[DBG$NMATCH(.INPUT_DESC, DBG$CS_RPG, 2)]:
      NOUN_NODE[DBG$ _NOUN_VALUE] = DBG$K_RPG;
[IF NOT .DBG$GL DEVELOPER[0] THEN FALSE ELSE
  DBG$NMATCH(.INPUT_DESC, DBG$CS_ADA, 2)]:
      NOUN_NODE[DBG$ _NOUN_VALUE] = DBG$K_ADA;
[DBG$NMATCH(.INPUT_DESC, DBG$CS_UNKNOWN, 3)]:
      NOUN_NODE[DBG$ _NOUN_VALUE] = DBG$K_UNKNOWN;
[OTHERWISE] :
      BEGIN
      IF DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)
      THEN
        .MESSAGE_VECT = DBG$NMAKE_ARG_VECT (DBG$_NEEDMORE)
      ELSE
        .MESSAGE_VECT =
          DBG$NSYNTAX_ERROR(DBG$NNEXT_WORD(.INPUT_DESC));
      RETURN ST$K_SEVERE;
      END;
      TES;
      END;
[DBG$NMATCH (.INPUT_DESC, DBG$CS_LOG, 2)] :
      BEGIN
      VERB_NODE [DBG$B_VERB_COMPOSITE] = SET_LOG;
      IF NOT DBG$NSAVE_FILESP (.INPUT_DESC,
        NOUN_NODE [DBG$ _NOUN_VALUE],
        .MESSAGE_VECT)
      THEN

```

```

: 777 0908 3
: 778 0909 3
: 779 0910 3
: 780 0911 3
: 781 0912 3
: 782 0913 3
: 783 0914 3
: 784 0915 3
: 785 0916 3
: 786 0917 3
: 787 0918 3
: 788 0919 3
: 789 0920 3
: 790 0921 3
: 791 0922 3
: 792 0923 3
: 793 0924 3
: 794 0925 3
: 795 0926 3
: 796 0927 3
: 797 0928 4
: 798 0929 4
: 799 0930 4
800 0931 4
801 0932 4
802 0933 4
803 0934 4
804 0935 4
805 0936 4
806 0937 4
807 0938 4
808 0939 4
809 0940 3
810 0941 4
811 0942 4
812 0943 4
813 0944 4
814 0945 4
815 0946 4
816 0947 4
817 0948 4
818 0949 4
819 0950 4
820 0951 4
821 0952 5
822 0953 5
823 0954 5
824 0955 5
825 0956 5
826 0957 5
827 0958 5
828 0959 5
829 0960 5
830 0961 5
831 0962 5
832 0963 6
: 833 0964 6

RETURN ST$K_SEVERE;
END;
[DBG$NMATCH (.INPUT_DESC, DBG$CS_MARGINS, 3)] :
BEGIN
LOCAL
LEFT_MARGIN,
RIGHT_MARGIN;

VERB_NODE[DBG$B_VERB_COMPOSITE] = SET_MARGINS;

! Legal forms of the SET MARGINS command are:
! 1. SET MARGINS l:r
! 2. SET MARGINS r
! 3. SET MARGINS l:
! 4. SET MARGINS :r
IF DBG$NMATCH (.INPUT_DESC, DBG$CS_COLON, 1)
THEN
BEGIN

! This must be case four, above
!
IF NOT DBG$NSAVE_DECIMAL_INTEGER(.INPUT_DESC,
RIGHT_MARGIN, .MESSAGE_VECT)
THEN
RETURN ST$K_SEVERE;
LEFT_MARGIN = .DBG$SRC_LEFT_MARGIN;
END
ELSE
BEGIN
IF NOT DBG$NSAVE_DECIMAL_INTEGER (.INPUT_DESC,
RIGHT_MARGIN, .MESSAGE_VECT)
THEN
RETURN ST$K_SEVERE;

! Look for colon
!
IF DBG$NMATCH (.INPUT_DESC, DBG$CS_COLON, 1)
THEN
BEGIN

! This must be case 1 or case 3 above
!
LEFT_MARGIN = .RIGHT_MARGIN;
IF DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)
THEN ! case 3
RIGHT_MARGIN = .DBG$SRC_RIGHT_MARGIN
ELSE ! case 1
BEGIN
IF NOT DBG$NSAVE_DECIMAL_INTEGER (.INPUT_DESC,
```



```

: 834      0965 6
: 835      0966 6
: 836      0967 6
: 837      0968 6
: 838      0969 6
: 839      0970 6
: 840      0971 6
: 841      0972 6
: 842      0973 6
: 843      0974 6
: 844      0975 6
: 845      0976 6
: 846      0977 6
: 847      0978 6
: 848      0979 6
: 849      0980 6
: 850      0981 6
: 851      0982 6
: 852      0983 6
: 853      0984 6
: 854      0985 6
: 855      0986 6
: 856      0987 6
: 857      0988 6
: 858      0989 6
: 859      0990 6
: 860      0991 6
: 861      0992 6
: 862      0993 6
: 863      0994 6
: 864      0995 6
: 865      0996 6
: 866      0997 6
: 867      0998 6
: 868      0999 6
: 869      1000 6
: 870      1001 6
: 871      1002 6
: 872      1003 6
: 873      1004 6
: 874      1005 6
: 875      1006 6
: 876      1007 6
: 877      1008 6
: 878      1009 6
: 879      1010 6
: 880      1011 6
: 881      1012 6
: 882      1013 6
: 883      1014 6
: 884      1015 6
: 885      1016 6
: 886      1017 6
: 887      1018 6
: 888      1019 6
: 889      1020 6
: 890      1021 6

      RIGHT_MARGIN, .MESSAGE_VECT)
      THEN
      RETURN ST$K_SEVERE;

      END;

      END

      ELSE
      ! case 2
      BEGIN
      LEFT_MARGIN = 1;
      END;

      END;

      ! Check that left margin is less than right margin
      !
      IF .LEFT_MARGIN GEQ .RIGHT_MARGIN
      THEN
      BEGIN
      DBG$NSHOW MARGINS();
      SIGNAL(DBG$_INVMA);
      END;

      ! Fill in the noun node
      !
      NOUN_NODE[DBG$L_NOUN_VALUE] = .LEFT_MARGIN;
      NOUN_NODE[DBG$L_NOUN_VALUE2] = .RIGHT_MARGIN;

      END;

      [DBG$NMATCH (.INPUT_DESC, DBG$CS_MAX_SOURCE_FILES, 3)] :
      BEGIN
      LOCAL
      MAX_FILES;
      VERB_NODE[DBG$B_VERB_COMPOSITE] = SET_MAX_SOURCE_FILES;
      IF NOT DBG$NSAVE_DECIMAL_INTEGER(
      .INPUT_DESC, MAX_FILES, .MESSAGE_VECT)
      THEN
      RETURN ST$K_SEVERE;
      NOUN_NODE[DBG$L_NOUN_VALUE] = .MAX_FILES;
      END;

      [DBG$NMATCH (.INPUT_DESC, DBG$CS_MODE, 1)] :
      BEGIN
      LOCAL
      LINK;

      BIND
      DBG$CS_BINARY          = UPLIT BYTE(%ASCIC 'BINARY'),
      DBG$CS_DECIMAL        = UPLIT BYTE(%ASCIC 'DECIMAL'),
      DBG$CS_DEFAULT        = UPLIT BYTE(%ASCIC 'DEFAULT'),
      DBG$CS_G_FLOAT        = UPLIT BYTE(%ASCIC 'G_FLOAT'),
```


DBGNSET
V04-000

I 15
16-Sep-1984 01:58:19 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:17:21 [DEBUG.SRC]DBGNSET.B32;1

Page 18
(3)

```
891 1022 3
892 1023 3
893 1024 3
894 1025 3
895 1026 3
896 1027 3
897 1028 3
898 1029 3
899 1030 3
900 1031 3
901 1032 3
902 1033 3
903 1034 3
904 1035 3
905 1036 3
906 1037 4
907 1038 4
908 1039 4
909 1040 4
910 1041 4
911 1042 4
912 1043 4
913 1044 4
914 1045 4
915 1046 4
916 1047 4
917 1048 4
918 1049 4
919 1050 4
920 1051 4
921 1052 4
922 1053 4
923 1054 4
924 1055 4
925 1056 4
926 1057 4
927 1058 4
928 1059 4
929 1060 4
930 1061 4
931 1062 4
932 1063 4
933 1064 4
934 1065 4
935 1066 4
936 1067 4
937 1068 4
938 1069 4
939 1070 4
940 1071 4
941 1072 4
942 1073 4
943 1074 4
944 1075 4
945 1076 4
946 1077 4
947 1078 4
```

```
DBG$CS_HEXADecimal = UPLIT BYTE(%ASCIC 'HEXADECIMAL'),
DBG$CS_KeYpAD = UPLIT BYTE(%ASCIC 'KEYPAD'),
DBG$CS_NOG FLOAT = UPLIT BYTE(%ASCIC 'NOG FLOAT'),
DBG$CS_NOKeYpAD = UPLIT BYTE(%ASCIC 'NOKEYPAD'),
DBG$CS_NOSCReeN = UPLIT BYTE(%ASCIC 'NOSCREEN'),
DBG$CS_NOSYMBOLS = UPLIT BYTE(%ASCIC 'NOSYMBOLIC'),
DBG$CS_OCTAL = UPLIT BYTE(%ASCIC 'OCTAL'),
DBG$CS_SCReeN = UPLIT BYTE(%ASCIC 'SCREEN'),
DBG$CS_SYMBOLS = UPLIT BYTE(%ASCIC 'SYMBOLIC');
```

```
VERB_NODE [DBG$B_VERB_COMPOSITE] = SET_MODE;
```

```
! Accept the mode switches
```

```
!
WHILE TRUE DO
BEGIN
```

```
SELECTONE TRUE OF
SET
```

```
[DBG$NMATCH (.INPUT_DESC, DBG$CS_BINARY, 1)] :
NOUN_NODE [DBG$[_NOUN_VALUE]] = MODE_BINARY;
```

```
[DBG$NMATCH (.INPUT_DESC, DBG$CS_DEFAULT, 3)] :
NOUN_NODE [DBG$[_NOUN_VALUE]] = MODE_DEFAULT;
```

```
[DBG$NMATCH (.INPUT_DESC, DBG$CS_DECIMAL, 1)] :
NOUN_NODE [DBG$[_NOUN_VALUE]] = MODE_DECIMAL;
```

```
[DBG$NMATCH (.INPUT_DESC, DBG$CS_HEXADecimal, 1)] :
NOUN_NODE [DBG$[_NOUN_VALUE]] = MODE_HEX;
```

```
[DBG$NMATCH (.INPUT_DESC, DBG$CS_KeYpAD, 1)] :
NOUN_NODE [DBG$[_NOUN_VALUE]] = MODE_KeYpAD;
```

```
[DBG$NMATCH (.INPUT_DESC, DBG$CS_NOKeYpAD, 3)] :
NOUN_NODE [DBG$[_NOUN_VALUE]] = MODE_NOKeYpAD;
```

```
[DBG$NMATCH (.INPUT_DESC, DBG$CS_NOSCReeN, 5)] :
NOUN_NODE [DBG$[_NOUN_VALUE]] = MODE_NOSCReeN;
```

```
[DBG$NMATCH (.INPUT_DESC, DBG$CS_NOSYMBOLS, 1)] :
NOUN_NODE [DBG$[_NOUN_VALUE]] = MODE_NOSYMBOLS;
```

```
[DBG$NMATCH (.INPUT_DESC, DBG$CS_OCTAL, 1)] :
NOUN_NODE [DBG$[_NOUN_VALUE]] = MODE_OCTAL;
```

```
[DBG$NMATCH (.INPUT_DESC, DBG$CS_SCReeN, 3)] :
NOUN_NODE [DBG$[_NOUN_VALUE]] = MODE_SCReeN;
```

```
[DBG$NMATCH (.INPUT_DESC, DBG$CS_SYMBOLS, 1)] :
NOUN_NODE [DBG$[_NOUN_VALUE]] = MODE_SYMBOLS;
```

```
[DBG$NMATCH (.INPUT_DESC, DBG$CS_G_FLOAT, 1)] :
NOUN_NODE [DBG$[_NOUN_VALUE]] = MODE_G_FLOAT;
```

```
[DBG$NMATCH (.INPUT_DESC, DBG$CS_NOG_FLOAT, 3)] :
```



```

: 948      1079  4      NOUN_NODE [DBG$NOUN_VALUE] = MODE_NOG_FLOAT;
: 949      1080  4
: 950      1081  4      [OTHERWISE] :
: 951      1082  4      BEGIN
: 952      1083  4      IF DBG$NMATCH(.INPUT_DESC, DBG$CS_CR, 1)
: 953      1084  4      THEN
: 954      1085  4      .MESSAGE_VECT = DBG$NMAKE_ARG_VECT(DBG$NEEDMORE)
: 955      1086  4
: 956      1087  4      ELSE
: 957      1088  4      .MESSAGE_VECT = DBG$NSYNTAX_ERROR(
: 958      1089  4      DBG$NNEXT_WORD(.INPUT_DESC));
: 959      1090  4
: 960      1091  4      RETURN ST$K_SEVERE;
: 961      1092  4      END;
: 962      1093  4
: 963      1094  4      TES;
: 964      1095  4
: 965      1096  4      ! Check for a comma to see if more input should follow
: 966      1097  4      !
: 967      1098  4      IF NOT DBG$NMATCH (.INPUT_DESC, DBG$CS_COMMA, 1)
: 968      1099  4      THEN
: 969      1100  4      EXITLOOP;
: 970      1101  4
: 971      1102  4
: 972      1103  4      ! More input should follow. Create another noun node and link
: 973      1104  4      !
: 974      1105  4      LINK = NOUN_NODE [DBG$NOUN_LINK];
: 975      1106  4      NOUN_NODE = DBG$GET_TEMP_MEM (DBG$K_NOUN_NODE_SIZE);
: 976      1107  4      .LINK = .NOUN_NODE;
: 977      1108  4
: 978      1109  4
: 979      1110  4      END;      ! End of loop
: 980      1111  4
: 981      1112  4      ! Place a 0 in the last noun link field
: 982      1113  4      !
: 983      1114  4      NOUN_NODE [DBG$NOUN_LINK] = 0;
: 984      1115  4
: 985      1116  4      END;
: 986      1117  4
: 987      1118  4      [DBG$NMATCH (.INPUT_DESC, DBG$CS_MODULE, 4)] :
: 988      1119  4      BEGIN
: 989      1120  4
: 990      1121  4      LOCAL
: 991      1122  4      ALL_FLAG,
: 992      1123  4      ALLOCATE_FLAG,
: 993      1124  4      NOWAIT_FLAG;
: 994      1125  4
: 995      1126  4
: 996      1127  4
: 997      1128  4      ! Initialize the flags for SET MODULE/ALL and SET MODULE/NOWAIT
: 998      1129  4      !
: 999      1130  4      ALL_FLAG = FALSE;
: 1000     1131  4      ALLOCATE_FLAG = FALSE;
: 1001     1132  4      NOWAIT_FLAG = FALSE;
: 1002     1133  4
: 1003     1134  4
: 1004     1135  4      ! Check for SET MODULE/ALL or SET MODULE/NOWAIT.
```

1005	1136	3
1006	1137	3
1007	1138	4
1008	1139	4
1009	1140	4
1010	1141	4
1011	1142	4
1012	1143	4
1013	1144	4
1014	1145	4
1015	1146	4
1016	1147	4
1017	1148	4
1018	1149	4
1019	1150	4
1020	1151	4
1021	1152	5
1022	1153	5
1023	1154	4
1024	1155	4
1025	1156	4
1026	1157	4
1027	1158	5
1028	1159	5
1029	1160	5
1030	1161	5
1031	1162	5
1032	1163	5
1033	1164	5
1034	1165	5
1035	1166	5
1036	1167	5
1037	1168	4
1038	1169	4
1039	1170	3
1040	1171	3
1041	1172	3
1042	1173	3
1043	1174	3
1044	1175	3
1045	1176	3
1046	1177	3
1047	1178	3
1048	1179	3
1049	1180	3
1050	1181	3
1051	1182	3
1052	1183	3
1053	1184	3
1054	1185	3
1055	1186	3
1056	1187	3
1057	1188	3
1058	1189	3
1059	1190	3
1060	1191	3
1061	1192	3

```
!
WHILE DBG$NMATCH (.INPUT_DESC, DBG$CS_SLASH, 1) DO
  BEGIN
  BIND
    DBG$CS_ALL = UPLIT BYTE (3, 'ALL'),
    DBG$CS_ALLOCATE = UPLIT BYTE (8, 'ALLOCATE'),
    DBG$CS_NOWAIT = UPLIT BYTE (6, 'NOWAIT');

    IF DBG$NMATCH (.INPUT_DESC, DBG$CS_ALLOCATE, 4)
    THEN
      ALLOCATE_FLAG = TRUE

    ELSE IF DBG$NMATCH (.INPUT_DESC, DBG$CS_ALL, 1)
    THEN
      ALL_FLAG = TRUE

    ELSE IF (IF NOT .DBG$GL DEVELOPER[0] THEN FALSE ELSE
      DBG$NMATCH (.INPUT_DESC, DBG$CS_NOWAIT, 1))
    THEN
      NOWAIT_FLAG = TRUE

    ELSE
      BEGIN
      IF DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)
      THEN
        .MESSAGE_VECT = DBG$NMAKE_ARG_VECT(DBG$NEEDMORE)

      ELSE
        .MESSAGE_VECT = DBG$NSYNTAX_ERROR(
          DBG$NNEXT_WORD(.INPUT_DESC));

      RETURN ST$K_SEVERE;
      END;

    END;
```

```
! Use the ADVERB_POINTER field to indicate whether /NOWAIT
! was specified and/or whether /ALLOCATE was specified.
! The codes are
! 0 - neither /NOWAIT or /ALLOCATE
! 1 - /NOWAIT, not /ALLOCATE
! 2 - not /NOWAIT, /ALLOCATE
! 3 - both were specified
```

```
VERB_NODE [DBG$L_VERB_ADVERB_PTR] = .NOWAIT_FLAG +
  2*.ALLOCATE_FLAG;
```

```
! If /ALL was specified, indicate this in the verb_composite node.
```

```
IF .ALL_FLAG
THEN
  VERB_NODE [DBG$B_VERB_COMPOSITE] = SET_MODULE_ALL
```

```
! Only pick up a module if /ALL was not specified.
```



```
1062 1193 3
1063 1194 3
1064 1195 3
1065 1196 4
1066 1197 4
1067 1198 4
1068 1199 4
1069 1200 4
1070 1201 4
1071 1202 4
1072 1203 4
1073 1204 4
1074 1205 4
1075 1206 4
1076 1207 4
1077 1208 5
1078 1209 5
1079 1210 5
1080 1211 5
1081 1212 5
1082 1213 5
1083 1214 5
1084 1215 5
1085 1216 5
1086 1217 6
1087 1218 6
1088 1219 6
1089 1220 6
1090 1221 6
1091 1222 6
1092 1223 6
1093 1224 6
1094 1225 6
1095 1226 6
1096 1227 6
1097 1228 6
1098 1229 6
1099 1230 6
1100 1231 7
1101 1232 6
1102 1233 6
1103 1234 6
1104 1235 6
1105 1236 6
1106 1237 6
1107 1238 6
1108 1239 6
1109 1240 6
1110 1241 6
1111 1242 6
1112 1243 6
1113 1244 6
1114 1245 6
1115 1246 6
1116 1247 6
1117 1248 6
1118 1249 6
```

```
! We have a module name list to parse.
ELSE
  BEGIN
    BIND
      DBG$CS_COMMA = UPLIT BYTE (1, dbg$k_comma);
    LOCAL
      LINK;          ! Temporary pointer

    ! Accept strings and commas
    WHILE TRUE DO
      BEGIN
        ! For language C, we do some fancy footwork to
        ! make sure we preserve the original casing of
        ! the identifier (since casing is significant
        ! in C).
        IF .dbg$gb_language EQL dbg$k_c
        THEN
          BEGIN
            MAP
              input_desc: REF dbg$stg_desc;
            LOCAL
              length,
              new_pointer: REF VECTOR [,BYTE],
              pointer,      ! Pointer to orig. command input
                           ! Pointer into input string
              stg_desc: dbg$stg_desc, ! String descriptor
              temp_ptr;

            pointer = .input_desc[dsc$a_pointer];
            length = .input_desc[dsc$a_length];
            IF (.pointer LSS .dbg$gl_upcase_command_ptr[0]) OR
                (.pointer GTR .dbg$gl_upcase_command_ptr[1])
            THEN
              $DBG_ERROR('DBGNSHOW\DBG$NPARSE_SET 10');

            ! We unfortunately have to allocate memory
            ! and copy strings in order to stuff a
            ! trailing carriage return at the end.
            new_pointer = dbg$get_tempmem((.length+3)/4);
            temp_ptr = (.pointer = .dbg$gl_upcase_command_ptr[0]) +
                      .dbg$gl_orig_command_ptr;
            CH$MOVE (.length, .temp_ptr, .new_pointer);
            new_pointer[.length-1] = dbg$k_car_return;

            ! Fill in the string descriptor.
            stg_desc[dsc$b_class] = dsc$k_class_s;
            stg_desc[dsc$b_dtype] = dsc$k_dtype_t;
            stg_desc[dsc$a_length] = .length;
```


DBGNSET
V04-000

M 15
16-Sep-1984 01:58:19 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:17:21 [DEBUG.SRC]DBGNSET.B32;1

Page 22
(3)

```
1119 1250 6
1120 1251 6
1121 1252 6
1122 1253 6
1123 1254 6
1124 1255 6
1125 1256 6
1126 1257 6
1127 1258 6
1128 1259 6
1129 1260 6
1130 1261 6
1131 1262 6
1132 1263 6
1133 1264 6
1134 1265 6
1135 1266 6
1136 1267 6
1137 1268 6
1138 1269 6
1139 1270 6
1140 1271 5
1141 1272 5
1142 1273 5
1143 1274 5
1144 1275 5
1145 1276 5
1146 1277 5
1147 1278 5
1148 1279 5
1149 1280 5
1150 1281 5
1151 1282 5
1152 1283 5
1153 1284 5
1154 1285 5
1155 1286 5
1156 1287 5
1157 1288 5
1158 1289 5
1159 1290 5
1160 1291 5
1161 1292 5
1162 1293 4
1163 1294 4
1164 1295 4
1165 1296 4
1166 1297 4
1167 1298 4
1168 1299 4
1169 1300 3
1170 1301 3
1171 1302 2
1172 1303 2
1173 1304 2
1174 1305 3
1175 1306 3
```

```
stg_desc[dsc$a_pointer] = .new_pointer;
stg_desc[dsc$l_pos] = 0;

! Pick up the symbol name.
!
IF NOT dbg$nsave_string( stg_desc,
                        NOUN_NODE [DBG$L_NOUN_VALUE],
                        .message_vect)
THEN
    RETURN sts$k_severe;

! Update the input descriptor.
input_desc[dsc$w_length] = .input_desc[dsc$w_length] -
    (.length - .stg_desc[dsc$w_length]);
input_desc[dsc$a_pointer] = .input_desc[dsc$a_pointer] +
    (.length - .stg_desc[dsc$w_length]);
END

! All other languages besides C ...
ELSE

    IF NOT DBG$NSAVE_STRING(.INPUT_DESC,
                           NOUN_NODE [DBG$L_NOUN_VALUE], .MESSAGE_VECT)
    THEN
        RETURN STS$K_SEVERE;

! Check for a comma
!
IF NOT DBG$NMATCH (.INPUT_DESC, DBG$CS_COMMA, 1)
THEN
    EXITLOOP;

! Create a new noun node to hold the next string
!
LINK = NOUN_NODE [DBG$L_NOUN_LINK];
NOUN_NODE = DBG$GET_TEMPMEM (DBG$K_NOUN_NODE_SIZE);
.LINK = .NOUN_NODE;

END; ! End of loop

! Place a zero in the last link field
!
NOUN_NODE [DBG$L_NOUN_LINK] = 0;
VERB_NODE [DBG$B_VERB_COMPOSITE] = SET_MODULE;
END;

END;

[DBG$NMATCH (.INPUT_DESC, DBG$CS_OUTPUT, 1)] :
BEGIN
```


DBGNSET
V04-000

N 15
16-Sep-1984 01:58:19
14-Sep-1984 12:17:21

VAX-11 Bliss-32 V4.0-742
[DEBUG.SRC]DBGNSET.B32;1

Page 23
(3)

```
1176 1307 3
1177 1308 3
1178 1309 3
1179 1310 3
1180 1311 3
1181 1312 3
1182 1313 3
1183 1314 3
1184 1315 3
1185 1316 3
1186 1317 3
1187 1318 3
1188 1319 3
1189 1320 3
1190 1321 3
1191 1322 3
1192 1323 3
1193 1324 3
1194 1325 3
1195 1326 3
1196 1327 3
1197 1328 3
1198 1329 3
1199 1330 3
1200 1331 3
1201 1332 3
1202 1333 3
1203 1334 3
1204 1335 3
1205 1336 3
1206 1337 3
1207 1338 3
1208 1339 4
1209 1340 4
1210 1341 4
1211 1342 4
1212 1343 4
1213 1344 4
1214 1345 4
1215 1346 4
1216 1347 4
1217 1348 4
1218 1349 4
1219 1350 4
1220 1351 4
1221 1352 4
1222 1353 4
1223 1354 4
1224 1355 4
1225 1356 4
1226 1357 4
1227 1358 4
1228 1359 4
1229 1360 4
1230 1361 4
1231 1362 4
1232 1363 4
```

```
LITERAL
  NOUN_LITERAL_LOG           = 1,
  NOUN_LITERAL_NOLOG         = 2,
  NOUN_LITERAL_SCREEN        = 3,
  NOUN_LITERAL_NOSCREEN      = 4,
  NOUN_LITERAL_TERMINAL       = 5,
  NOUN_LITERAL_NOTERMINAL     = 6,
  NOUN_LITERAL_VERIFY         = 7,
  NOUN_LITERAL_NOVERIFY      = 8;

BIND
  DBG$CS_LOG                  = UPLIT BYTE (%ASCIC 'LOG'),
  DBG$CS_NOLOG                 = UPLIT BYTE (%ASCIC 'NOLOG'),
  DBG$CS_SCREEN_LOG           = UPLIT BYTE (%ASCIC 'SCREEN LOG'),
  DBG$CS_NOSCREEN_LOG         = UPLIT BYTE (%ASCIC 'NOSCREEN LOG'),
  DBG$CS_TERMINAL              = UPLIT BYTE (%ASCIC 'TERMINAL'),
  DBG$CS_NOTERMINAL            = UPLIT BYTE (%ASCIC 'NOTERMINAL'),
  DBG$CS_VERIFY                = UPLIT BYTE (%ASCIC 'VERIFY'),
  DBG$CS_NOVERIFY              = UPLIT BYTE (%ASCIC 'NOVERIFY');

LOCAL
  LINK;                        ! Used as a link pointer

! Set the verb composite.
VERB_NODE [DBG$B_VERB_COMPOSITE] = SET_OUTPUT;

! Accept the SET OUTPUT xxx objects.
WHILE TRUE DO
  BEGIN
    SELECTONE TRUE OF
      SET
        [DBG$NMATCH (.INPUT_DESC, DBG$CS_LOG, 1)] :
          NOUN_NODE [DBG$[_NOUN_VALUE]] = NOUN_LITERAL_LOG;
        [DBG$NMATCH (.INPUT_DESC, DBG$CS_NOLOG, 3)] :
          NOUN_NODE [DBG$[_NOUN_VALUE]] = NOUN_LITERAL_NOLOG;
        [DBG$NMATCH (.INPUT_DESC, DBG$CS_SCREEN_LOG, 1)] :
          NOUN_NODE [DBG$[_NOUN_VALUE]] = NOUN_LITERAL_SCREEN;
        [DBG$NMATCH (.INPUT_DESC, DBG$CS_NOSCREEN_LOG, 3)] :
          NOUN_NODE [DBG$[_NOUN_VALUE]] = NOUN_LITERAL_NOSCREEN;
        [DBG$NMATCH (.INPUT_DESC, DBG$CS_TERMINAL, 1)] :
          NOUN_NODE [DBG$[_NOUN_VALUE]] = NOUN_LITERAL_TERMINAL;
        [DBG$NMATCH (.INPUT_DESC, DBG$CS_NOTERMINAL, 3)] :
          NOUN_NODE [DBG$[_NOUN_VALUE]] = NOUN_LITERAL_NOTERMINAL;
        [DBG$NMATCH (.INPUT_DESC, DBG$CS_VERIFY, 1)] :
          NOUN_NODE [DBG$[_NOUN_VALUE]] = NOUN_LITERAL_VERIFY;
```



```
: 1233      1364  4      [DBG$NMATCH (.INPUT_DESC, DBG$CS_NOVERIFY, 3)] :
: 1234      1365  4      NOUN_NODE [DBG$C_NOUN_VALUE] = NOUN_LITERAL_NOVERIFY;
: 1235      1366  4
: 1236      1367  4
: 1237      1368  5      [OTHERWISE] :
: 1238      1369  5      BEGIN
: 1239      1370  5      IF DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)
: 1240      1371  5      THEN
: 1241      1372  5      .MESSAGE_VECT = DBG$NMAKE_ARG_VECT (DBG$NEEDMORE)
: 1242      1373  5
: 1243      1374  5      ELSE
: 1244      1375  5      .MESSAGE_VECT = DBG$NSYNTAX_ERROR (DBG$NNEXT_WORD
: 1245      1376  5      (.INPUT_DESC));
: 1246      1377  4      RETURN ST$K_SEVERE;
: 1247      1378  4      END;
: 1248      1379  4      TES;
: 1249      1380  4
: 1250      1381  4
: 1251      1382  4      ! Look for a comma and exit the loop if none found
: 1252      1383  4
: 1253      1384  4      IF NOT DBG$NMATCH (.INPUT_DESC, DBG$CS_COMMA, 1)
: 1254      1385  4      THEN
: 1255      1386  4      EXITLOOP;
: 1256      1387  4
: 1257      1388  4
: 1258      1389  4      ! Since we expect more input, create another noun node
: 1259      1390  4
: 1260      1391  4      LINK = NOUN_NODE [DBG$L_NOUN_LINK];
: 1261      1392  4      NOUN_NODE = DBG$GET_TEMP_MEM (DBG$K_NOUN_NODE_SIZE);
: 1262      1393  4      .LINK = .NOUN_NODE;
: 1263      1394  4
: 1264      1395  3      END;                                ! End of loop
: 1265      1396  3
: 1266      1397  3
: 1267      1398  3      ! Put a zero in the last link field of the noun node chain
: 1268      1399  3
: 1269      1400  3      NOUN_NODE [DBG$L_NOUN_LINK] = 0;
: 1270      1401  3
: 1271      1402  2      END;
: 1272      1403  2
: 1273      1404  2      [DBG$NMATCH (.INPUT_DESC, DBG$CS_RADIX, 1)] :
: 1274      1405  3      BEGIN
: 1275      1406  3      BIND
: 1276      1407  3      DBG$CS_BINARY          = UPLIT BYTE(%ASCIC 'BINARY'),
: 1277      1408  3      DBG$CS_DECIMAL          = UPLIT BYTE(%ASCIC 'DECIMAL'),
: 1278      1409  3      DBG$CS_DEFAULT          = UPLIT BYTE(%ASCIC 'DEFAULT'),
: 1279      1410  3      DBG$CS_HEXADECIMAL       = UPLIT BYTE(%ASCIC 'HEXADECIMAL'),
: 1280      1411  3      DBG$CS_INPUT           = UPLIT BYTE(%ASCIC 'INPUT'),
: 1281      1412  3      DBG$CS_OCTAL            = UPLIT BYTE(%ASCIC 'OCTAL'),
: 1282      1413  3      DBG$CS_OUTPUT          = UPLIT BYTE(%ASCIC 'OUTPUT'),
: 1283      1414  3      DBG$CS_OVERRIDE        = UPLIT BYTE(%ASCIC 'OVERRIDE');
: 1284      1415  3
: 1285      1416  3      ! Check for the /OVERRIDE switch. Then fill in the verb
: 1286      1417  3      composite field accordingly. Also check for /INPUT or
: 1287      1418  3      /OUTPUT.
: 1288      1419  3
: 1289      1420  3      VERB_NODE[DBG$B_VERB_COMPOSITE] = SET_RADIX;
```



```
: 1290      1421      3
: 1291      1422      3
: 1292      1423      3
: 1293      1424      3
: 1294      1425      3
: 1295      1426      3
: 1296      1427      3
: 1297      1428      3
: 1298      1429      3
: 1299      1430      3
: 1300      1431      3
: 1301      1432      3
: 1302      1433      3
: 1303      1434      3
: 1304      1435      4
: 1305      1436      4
: 1306      1437      4
: 1307      1438      4
: 1308      1439      4
: 1309      1440      4
: 1310      1441      4
: 1311      1442      4
: 1312      1443      5
: 1313      1444      5
: 1314      1445      5
: 1315      1446      5
: 1316      1447      5
: 1317      1448      5
: 1318      1449      5
: 1319      1450      4
: 1320      1451      3
: 1321      1452      3
: 1322      1453      3
: 1323      1454      3
: 1324      1455      3
: 1325      1456      3
: 1326      1457      3
: 1327      1458      3
: 1328      1459      3
: 1329      1460      3
: 1330      1461      3
: 1331      1462      3
: 1332      1463      3
: 1333      1464      3
: 1334      1465      3
: 1335      1466      3
: 1336      1467      3
: 1337      1468      3
: 1338      1469      3
: 1339      1470      3
: 1340      1471      3
: 1341      1472      3
: 1342      1473      3
: 1343      1474      3
: 1344      1475      4
: 1345      1476      4
: 1346      1477      4
```

```
WHILE DBG$NMATCH (.INPUT_DESC, DBG$CS_SLASH, 1) DO
  SELECTONE TRUE OF
    SET
      [DBG$NMATCH (.INPUT_DESC, DBG$CS_INPUT, 1)]:
        NOUN_NODE[DBG$L_NOUN_VALUE2] = 1;
      [DBG$NMATCH (.INPUT_DESC, DBG$CS_OUTPUT, 2)]:
        NOUN_NODE[DBG$L_NOUN_VALUE2] = 2;
      [DBG$NMATCH (.INPUT_DESC, DBG$CS_OVERRIDE, 2)]:
        VERB_NODE [DBG$B_VERB_COMPOSITE] = SET_RADIX_OVERRIDE;
      [OTHERWISE]:
        BEGIN
          LOCAL
            CS: REF VECTOR[BYTE],
            STG_DESC: DBG$STG_DESC;
          IF DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)
            THEN
              SIGNAL(DBG$_NEEDMORE)
            ELSE
              BEGIN
                CS = DBG$NNEXT_WORD(.INPUT_DESC);
                STG_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
                STG_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
                STG_DESC[DSC$W_LENGTH] = .CS[0];
                STG_DESC[DSC$A_POINTER] = CS[1];
                SIGNAL(DBG$_SYNTAX, 1, STG_DESC);
              END;
            END;
        END;
  TES;
! Accept the radix argument.
SELECTONE TRUE OF
  SET
    [DBG$NMATCH (.INPUT_DESC, DBG$CS_BINARY, 1)]:
      NOUN_NODE [DBG$_NOUN_VALUE] = MODE_BINARY;
    [DBG$NMATCH (.INPUT_DESC, DBG$CS_DEFAULT, 3)]:
      NOUN_NODE [DBG$_NOUN_VALUE] = MODE_DEFAULT;
    [DBG$NMATCH (.INPUT_DESC, DBG$CS_DECIMAL, 1)]:
      NOUN_NODE [DBG$_NOUN_VALUE] = MODE_DECIMAL;
    [DBG$NMATCH (.INPUT_DESC, DBG$CS_HEXADECEMAL, 1)]:
      NOUN_NODE [DBG$_NOUN_VALUE] = MODE_HEX;
    [DBG$NMATCH (.INPUT_DESC, DBG$CS_OCTAL, 1)]:
      NOUN_NODE [DBG$_NOUN_VALUE] = MODE_OCTAL;
    [OTHERWISE]:
      BEGIN
        IF DBG$NMATCH(.INPUT_DESC, DBG$CS_CR, 1)
          THEN
```



```
1347      .MESSAGE_VECT = DBG$NMAKE_ARG_VECT(DBG$_NEEDMORE)
1348
1349      ELSE
1350      .MESSAGE_VECT = DBG$NSYNTAX_ERROR(
1351      DBG$NNEXT_WORD(.INPUT_DESC));
1352      RETURN ST$K_SEVERE;
1353      END;
1354
1355      TES;
1356      END;
1357
1358      [DBG$NMATCH (.INPUT_DESC, DBG$CS_SCOPE, 2)] :
1359      BEGIN
1360      VERB_NODE [DBG$B_VERB_COMPOSITE] = SET_SCOPE;
1361
1362      ! Fill in 1 to the ADVERB field if SET SCOPE/MODULE was
1363      ! specified; 0 otherwise.
1364
1365      IF DBG$NMATCH (.INPUT_DESC, DBG$CS_SLASH, 1)
1366      THEN
1367      IF DBG$NMATCH (.INPUT_DESC, DBG$CS_MODULE, 1)
1368      THEN
1369      VERB_NODE[DBG$_VERB_ADVERB_PTR] = 1
1370      ELSE
1371      SIGNAL(DBG$_SETSCOMOD, 1, DBG$NNEXT_WORD(.INPUT_DESC))
1372      ELSE
1373      VERB_NODE[DBG$L_VERB_ADVERB_PTR] = 0;
1374
1375      IF NOT DBG$NPARSE_SCOPE_LIST(.INPUT_DESC,
1376      NOUN_NODE[DBG$L_NOUN_VALUE], .MESSAGE_VECT)
1377      THEN
1378      RETURN ST$K_SEVERE;
1379      END;
1380
1381      [DBG$NMATCH (.INPUT_DESC, DBG$CS_SEARCH, 2)] :
1382      BEGIN
1383      LITERAL
1384      NOUN_LITERAL_ALL = 1,
1385      NOUN_LITERAL_NEXT = 2,
1386      NOUN_LITERAL_STRING = 3,
1387      NOUN_LITERAL_IDENT = 4;
1388
1389      BIND
1390      DBG$CS_ALL = UPLIT BYTE (3, 'ALL'),
1391      DBG$CS_NEXT = UPLIT BYTE (4, 'NEXT'),
1392      DBG$CS_STRING = UPLIT BYTE (6, 'STRING'),
1393      DBG$CS_IDENT = UPLIT BYTE(10, 'IDENTIFIER');
1394
1395      LOCAL
1396      LINK; ! Used as a link pointer
1397
1398      ! Set up the verb composite
1399      !
```


1404 1535 3
1405 1536 3
1406 1537 3
1407 1538 3
1408 1539 3
1409 1540 4
1410 1541 4
1411 1542 4
1412 1543 4
1413 1544 4
1414 1545 4
1415 1546 4
1416 1547 4
1417 1548 4
1418 1549 4
1419 1550 4
1420 1551 4
1421 1552 4
1422 1553 4
1423 1554 4
1424 1555 4
1425 1556 4
1426 1557 5
1427 1558 5
1428 1559 5
1429 1560 5
1430 1561 5
1431 1562 5
1432 1563 5
1433 1564 5
1434 1565 5
1435 1566 5
1436 1567 4
1437 1568 4
1438 1569 4
1439 1570 4
1440 1571 4
1441 1572 4
1442 1573 4
1443 1574 4
1444 1575 4
1445 1576 4
1446 1577 4
1447 1578 4
1448 1579 4
1449 1580 4
1450 1581 4
1451 1582 4
1452 1583 4
1453 1584 4
1454 1585 3
1455 1586 3
1456 1587 3
1457 1588 3
1458 1589 3
1459 1590 3
1460 1591 3

```
VERB_NODE[DBG$B_VERB_COMPOSITE] = SET_SEARCH;
! Accept the SET SEARCH xxx objects
!
WHILE TRUE DO
  BEGIN
    SELECTONE TRUE OF
      SET
        [DBG$NMATCH (.INPUT_DESC, DBG$CS_ALL, 1)] :
          NOUN_NODE [DBG$_NOUN_VALUE] = NOUN_LITERAL_ALL;
        [DBG$NMATCH (.INPUT_DESC, DBG$CS_IDENT, 1)] :
          NOUN_NODE [DBG$_NOUN_VALUE] = NOUN_LITERAL_IDENT;
        [DBG$NMATCH (.INPUT_DESC, DBG$CS_NEXT, 1)] :
          NOUN_NODE [DBG$_NOUN_VALUE] = NOUN_LITERAL_NEXT;
        [DBG$NMATCH (.INPUT_DESC, DBG$CS_STRING, 1)] :
          NOUN_NODE [DBG$_NOUN_VALUE] = NOUN_LITERAL_STRING;
        [OTHERWISE] :
          BEGIN
            IF DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)
              THEN
                .MESSAGE_VECT = DBG$NMAKE_ARG_VECT (DBG$_NEEDMORE)
              ELSE
                .MESSAGE_VECT = DBG$NSYNTAX_ERROR(
                  DBG$NNEXT_WORD(.INPUT_DESC));

            RETURN ST$K_SEVERE;
          END;
      TES;

! Look for comma and exit the loop if none found.
!
IF NOT DBG$NMATCH (.INPUT_DESC, DBG$CS_COMMA, 1)
  THEN
    EXITLOOP;

! Since we expect more input, create another noun node.
!
LINK = NOUN_NODE[DBG$L_NOUN_LINK];
NOUN_NODE = DBG$GET_TEMPMEM (DBG$K_NOUN_NODE_SIZE);
.LINK = .NOUN_NODE;

END;                                ! End of loop

! Put a zero in the last link field of the noun node chain.
!
NOUN_NODE [DBG$L_NOUN_LINK] = 0;
```



```

: 1461
: 1462
: 1463
: 1464
: 1465
: 1466
: 1467
: 1468
: 1469
: 1470
: 1471
: 1472
: 1473
: 1474
: 1475
: 1476
: 1477
: 1478
: 1479
: 1480
: 1481
: 1482
: 1483
: 1484
: 1485
: 1486
: 1487
: 1488
: 1489
: 1490
: 1491
: 1492
: 1493
: 1494
: 1495
: 1496
: 1497
: 1498
: 1499
: 1500
: 1501
: 1502
: 1503
: 1504
: 1505
: 1506
: 1507
: 1508
: 1509
: 1510
: 1511
: 1512
: 1513
: 1514
: 1515
: 1516
: 1517

```

```

1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648

```

```

END;

! Parse the SET SOURCE command.
[DBG$NMATCH (.INPUT_DESC, DBG$CS_SOURCE, 2)] :
BEGIN
  VERB_NODE[DBG$B_VERB_COMPOSITE] = SET_SOURCE;

! Check for SET SOURCE/MODULE=dir-list
! IF DBG$NMATCH (.INPUT_DESC, DBG$CS_SLASH, 1)
THEN
  BEGIN
    LOCAL
      MODNAMEPTR;

    BIND
      DBG$CS_MODULE = UPLIT BYTE (6, 'MODULE');

! Read the string MODULE.
! IF NOT DBG$NMATCH (.INPUT_DESC, DBG$CS_MODULE, 4)
THEN
  BEGIN
    .MESSAGE_VECT = DBG$NSYNTAX_ERROR(
      DBG$NNEXT_WORD(.INPUT_DESC));
    RETURN ST$K_SEVERE;
  END;

! Read the = sign.
! IF NOT DBG$NMATCH (.INPUT_DESC, DBG$CS_EQUAL, 1)
THEN
  BEGIN
    .MESSAGE_VECT = DBG$NSYNTAX_ERROR(
      DBG$NNEXT_WORD(.INPUT_DESC));
    RETURN ST$K_SEVERE;
  END;

! Read the module name.
! IF NOT DBG$NSAVE_STRING(.INPUT_DESC, MODNAMEPTR, .MESSAGE_VECT)
THEN
  RETURN ST$K_SEVERE;

! Convert the module name into an rst pointer and
! save into the adjective field of the noun node.
NOUN_NODE[DBG$L_ADJECTIVE_PTR] =
  DBG$STA_GETSourcemod(.MODNAMEPTR);

```



```
1518 1649 4
1519 1650 4
1520 1651 4
1521 1652 4
1522 1653 4
1523 1654 4
1524 1655 4
1525 1656 3
1526 1657 3
1527 1658 3
1528 1659 3
1529 1660 4
1530 1661 4
1531 1662 3
1532 1663 3
1533 1664 3
1534 1665 3
1535 1666 3
1536 1667 3
1537 1668 3
1538 1669 3
1539 1670 3
1540 1671 3
1541 1672 3
1542 1673 3
1543 1674 3
1544 1675 2
1545 1676 2
1546 1677 2
1547 1678 3
1548 1679 3
1549 1680 3
1550 1681 3
1551 1682 2
1552 1683 2
1553 1684 2
1554 1685 2
1555 1686 2
1556 1687 2
1557 1688 3
1558 1689 3
1559 1690 3
1560 1691 2
1561 1692 2
1562 1693 2
1563 1694 2
1564 1695 2
1565 1696 2
1566 1697 3
1567 1698 3
1568 1699 3
1569 1700 2
1570 1701 2
1571 1702 2
1572 1703 2
1573 1704 2
1574 1705 2

! If the above routine returns zero then the user has
! entered an invalid module.
IF .NOUN_NODE[DBG$$_ADJECTIVE_PTR] EQL 0
THEN
  BEGIN
    .MESSAGE_VECT = DBG$$_MAKE_ARG_VECT(
      DBG$$_NOSUCHMODU, 1, .MODNAMEPTR);
    RETURN ST$$_SEVERE;
  END;
END;

! SET SOURCE/MODULE=

! Now read the directory list.
IF NOT DBG$$_GET_DIR_LIST (.INPUT_DESC,
  NOUN_NODE[DBG$$_NOUN_VALUE], .MESSAGE_VECT)
THEN
  RETURN ST$$_SEVERE;
END;

! SET SOURCE

! Parse the SET STEP command.
[DBG$$_MATCH (.INPUT_DESC, DBG$$_CS_STEP, 1)] :
  BEGIN
    VERB_NODE [DBG$$_B_VERB_COMPOSITE] = EVENT$$_SET_STEP;
    RETURN DBG$$_EVENT_SYNTAX (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT);
  END;

! Parse the SET TASK command.
[IF NOT .DBG$$_GL_DEVELOPER[0] THEN FALSE ELSE
  DBG$$_MATCH (.INPUT_DESC, DBG$$_CS_TASK, 2)]:
  BEGIN
    VERB_NODE[DBG$$_B_VERB_COMPOSITE] = SET_TASK;
    DBG$$_PARSE_SET_TASK(.INPUT_DESC, .VERB_NODE);
  END;

! Parse the SET TERMINAL command.
[DBG$$_MATCH(.INPUT_DESC, DBG$$_CS_TERMINAL, 4)]:
  BEGIN
    VERB_NODE[DBG$$_B_VERB_COMPOSITE] = SET_TERMINAL;
    DBG$$_SCR_PARSE_SETTERM_CMD(.INPUT_DESC, .VERB_NODE);
  END;

! Parse the SET TRACE command.
[DBG$$_MATCH(.INPUT_DESC, DBG$$_CS_TRACE, 1)]:
```



```
: 1575 1706 3
: 1576 1707 3
: 1577 1708 3
: 1578 1709 3
: 1579 1710 3
: 1580 1711 3
: 1581 1712 3
: 1582 1713 3
: 1583 1714 3
: 1584 1715 3
: 1585 1716 3
: 1586 1717 3
: 1587 1718 3
: 1588 1719 3
: 1589 1720 3
: 1590 1721 3
: 1591 1722 3
: 1592 1723 3
: 1593 1724 3
: 1594 1725 3
: 1595 1726 3
: 1596 1727 3
: 1597 1728 3
: 1598 1729 3
: 1599 1730 3
: 1600 1731 3
: 1601 1732 3
: 1602 1733 3
: 1603 1734 3
: 1604 1735 3
: 1605 1736 3
: 1606 1737 3
: 1607 1738 3
: 1608 1739 3
: 1609 1740 3
: 1610 1741 3
: 1611 1742 3
: 1612 1743 3
: 1613 1744 3
: 1614 1745 3
: 1615 1746 3
: 1616 1747 3
: 1617 1748 3
: 1618 1749 3
: 1619 1750 3
: 1620 1751 3
: 1621 1752 3
: 1622 1753 4
: 1623 1754 4
: 1624 1755 4
: 1625 1756 4
: 1626 1757 4
: 1627 1758 4
: 1628 1759 5
: 1629 1760 5
: 1630 1761 6
: 1631 1762 6
```

```
BEGIN
VERB NODE [DBG$B_VERB_COMPOSITE] = SET_TRACE;
RETURN DBG$EVENT_SYNTAX (.INPUT_DESC,
                          .VERB_NODE,
                          .MESSAGE_VECT
                          );
```

```
END;
```

```
[dbg$match (.input_desc, dbg$cs_type, 2)] : ! SET TYPE or SET TYPE/OVERRIDE
```

```
BEGIN
LOCAL
```

```
LENGTH_NODE : REF dbg$noun_node;
```

```
BIND
```

```
DBG$CS_AC      = UPLIT BYTE (2, 'AC'),
DBG$CS_AD      = UPLIT BYTE (2, 'AD'),
DBG$CS_ASCIC   = UPLIT BYTE (5, 'ASCIC'),
DBG$CS_ASCID   = UPLIT BYTE (5, 'ASCID'),
DBG$CS_ASCII   = UPLIT BYTE (5, 'ASCII'),
DBG$CS_ASCIZ   = UPLIT BYTE (5, 'ASCIZ'),
DBG$CS_AZ      = UPLIT BYTE (2, 'AZ'),
DBG$CS_BYTE    = UPLIT BYTE (4, 'BYTE'),
DBG$CS_COLON   = UPLIT BYTE (1, dbg$k_colon),
DBG$CS_D_FLOAT = UPLIT BYTE (7, 'D_FLOAT'),
DBG$CS_FLOAT   = UPLIT BYTE (5, 'FLOAT'),
DBG$CS_F_FLOAT = UPLIT BYTE (7, 'F_FLOAT'),
DBG$CS_G_FLOAT = UPLIT BYTE (7, 'G_FLOAT'),
DBG$CS_H_FLOAT = UPLIT BYTE (7, 'H_FLOAT'),
DBG$CS_PACKED  = UPLIT BYTE (6, 'PACKED'),
DBG$CS_INSTRUCTION = UPLIT BYTE (11, 'INSTRUCTION'),
DBG$CS_DATE_TIME = UPLIT BYTE (9, 'DATE TIME'),
DBG$CS_LONGWORD = UPLIT BYTE (8, 'LONGWORD'),
DBG$CS_OCTAWORD = UPLIT BYTE (8, 'OCTAWORD'),
DBG$CS_OVERRIDE = UPLIT BYTE (8, 'OVERRIDE'),
DBG$CS_QUADWORD = UPLIT BYTE (8, 'QUADWORD'),
DBG$CS_WORD    = UPLIT BYTE (4, 'WORD');
```

```
! Create and link the length node (the second noun node).
```

```
length_node = dbg$get_tempmem (dbg$k_noun_node_size);
noun_node [dbg$l_noun_link] = .length_node;
```

```
! Look for slash
```

```
IF dbg$match (.input_desc, dbg$cs_slash, 1)
```

```
THEN
```

```
BEGIN
```

```
! Must find 'override'
```

```
IF NOT dbg$match (.input_desc, dbg$cs_override, 1)
```

```
THEN
```

```
BEGIN
```

```
.message_vect =
```

```
(IF dbg$match (.input_desc, dbg$cs_cr, 1)
```

```
THEN
```



```

: 1632      1763  6      dbg$make_arg_vect (dbg$_needmore)
: 1633      1764  6      ELSE
: 1634      1765  5      dbg$nsyntax_error (dbg$next_word (.input_desc));
: 1635      1766  5      RETURN sts$k_severe;
: 1636      1767  4      END;
: 1637      1768  4
: 1638      1769  4      ! Set the verb composite
: 1639      1770  4      !
: 1640      1771  4      verb_node [dbg$b_verb_composite] = set_type_override;
: 1641      1772  4      END
: 1642      1773  4
: 1643      1774  4      ELSE
: 1644      1775  4      verb_node [dbg$b_verb_composite] = set_type;
: 1645      1776  4
: 1646      1777  4      ! The verb composite has been set. We must recognize the type specified.
: 1647      1778  4      !
: 1648      1779  4      SELECTONE true
: 1649      1780  4      OF
: 1650      1781  4      SET
: 1651      1782  4
: 1652      1783  4      [dbg$match (.input_desc, dbg$cs_ascic, 5),
: 1653      1784  4      dbg$match (.input_desc, dbg$cs_ac, 2)] :
: 1654      1785  4      BEGIN
: 1655      1786  4      noun_node [dbg$l_noun_value] = dsc$k_dtype_ac;
: 1656      1787  4      length_node [dbg$l_noun_value] = 0;
: 1657      1788  4      END;
: 1658      1789  4
: 1659      1790  4      [dbg$match (.input_desc, dbg$cs_ascid, 5) OR
: 1660      1791  4      dbg$match (.input_desc, dbg$cs_ad, 2)] :
: 1661      1792  4      BEGIN
: 1662      1793  4      noun_node [dbg$l_noun_value] = dsc$k_dtype_ad;
: 1663      1794  4      length_node [dbg$l_noun_value] = 8;
: 1664      1795  4      END;
: 1665      1796  4
: 1666      1797  4      [dbg$match (.input_desc, dbg$cs_packed, 1)] :
: 1667      1798  4      BEGIN
: 1668      1799  4      noun_node [dbg$l_noun_value] = dsc$k_dtype_p;
: 1669      1800  4
: 1670      1801  4      ! Accept the ':' and integer. If there is no colon, use
: 1671      1802  4      ! a length or default of 10.
: 1672      1803  4      !
: 1673      1804  4      IF NOT dbg$match (.input_desc, dbg$cs_colon, 1)
: 1674      1805  4      THEN
: 1675      1806  4      SIGNAL (dbg$_pacsizreq)
: 1676      1807  4      ELSE
: 1677      1808  4      BEGIN
: 1678      1809  4      ! We found the colon. Accept the integer
: 1679      1810  4      !
: 1680      1811  4      IF NOT dbg$nsave_integer (.input_desc,
: 1681      1812  4      length_node [dbg$l_noun_value])
: 1682      1813  4      THEN
: 1683      1814  4      RETURN sts$k_severe;
: 1684      1815  4
: 1685      1816  4      ! Check to see that the length is within 0 and 31
: 1686      1817  4      !
: 1687      1818  4      IF (.length_node [dbg$l_noun_value] LSS 0) OR (.length_node [dbg$l_noun_value] GTR
: 1688      1819  6
```


:	1689	1820	5
:	1690	1821	5
:	1691	1822	4
:	1692	1823	3
:	1693	1824	3
:	1694	1825	3
:	1695	1826	4
:	1696	1827	4
:	1697	1828	4
:	1698	1829	4
:	1699	1830	4
:	1700	1831	4
:	1701	1832	4
:	1702	1833	4
:	1703	1834	4
:	1704	1835	4
:	1705	1836	4
:	1706	1837	4
:	1707	1838	5
:	1708	1839	5
:	1709	1840	5
:	1710	1841	5
:	1711	1842	5
:	1712	1843	5
:	1713	1844	5
:	1714	1845	5
:	1715	1846	5
:	1716	1847	5
:	1717	1848	5
:	1718	1849	5
:	1719	1850	5
:	1720	1851	6
:	1721	1852	6
:	1722	1853	6
:	1723	1854	6
:	1724	1855	6
:	1725	1856	6
:	1726	1857	6
:	1727	1858	6
:	1728	1859	6
:	1729	1860	6
:	1730	1861	6
:	1731	1862	6
:	1732	1863	5
:	1733	1864	4
:	1734	1865	3
:	1735	1866	3
:	1736	1867	3
:	1737	1868	3
:	1738	1869	4
:	1739	1870	4
:	1740	1871	4
:	1741	1872	3
:	1742	1873	3
:	1743	1874	3
:	1744	1875	4
:	1745	1876	4

```
THEN
    SIGNAL(dbg$_illpacsiz, 1, .length_node [dbg$_noun_value]);
END;
END;

[dbg$_nmatch (.input_desc, dbg$_cs_ascii, 1)] :
BEGIN
    noun_node [dbg$_noun_value] = dsc$_k_dtype_t;

    ! There may be a colon followed by integer. If not,
    ! we use a default length of 4.
    length_node [dbg$_noun_value] = 4;

    ! See if a colon follows
    IF dbg$_nmatch (.input_desc, dbg$_cs_colon, 1)
    THEN
        BEGIN
            ! Obtain the integer.
            IF NOT dbg$_nsave_integer (.input_desc,
                                     length_node [dbg$_noun_value])
            THEN
                RETURN sts$_k_severe;

            ! Check for a zero length
            IF .length_node [dbg$_noun_value] EQL 0
            THEN
                BEGIN
                    ! Set up for an error message
                    OWN
                    ZERO_DESC : dbg$_stg_desc;
                    zero_desc [dsc$_w_length] = 1;
                    zero_desc [dsc$_a_pointer] = UPLIT BYTE ('0');
                    .message_vect = dbg$_nmake_arg_vect (dbg$_invnumber,
                                                         1,
                                                         zero_desc);

                    RETURN sts$_k_severe;
                END;
            END;
        END;
    END;

[dbg$_nmatch (.input_desc, dbg$_cs_asciz, 5),
 dbg$_nmatch (.input_desc, dbg$_cs_az, 2)] :
BEGIN
    noun_node [dbg$_noun_value] = dsc$_k_dtype_az;
    length_node [dbg$_noun_value] = 0;
END;

[dbg$_nmatch (.input_desc, dbg$_cs_byte, 1)] :
BEGIN
    noun_node [dbg$_noun_value] = dsc$_k_dtype_b;
```


:	1746	1877	4	length_node [dbg\$l_noun_value] = 1;
:	1747	1878	3	END;
:	1748	1879	3	
:	1749	1880	3	[dbg\$match (.input_desc, dbg\$cs_date_time, 2)] :
:	1750	1881	4	BEGIN
:	1751	1882	4	noun_node [dbg\$l_noun_value] = dsc\$k_dtype_adt;
:	1752	1883	4	length_node [dbg\$l_noun_value] = 8;
:	1753	1884	3	END;
:	1754	1885	3	
:	1755	1886	3	[dbg\$match (.input_desc, dbg\$cs_d_float, 1)] :
:	1756	1887	4	BEGIN
:	1757	1888	4	noun_node [dbg\$l_noun_value] = dsc\$k_dtype_d;
:	1758	1889	4	length_node [dbg\$l_noun_value] = 8;
:	1759	1890	3	END;
:	1760	1891	3	
:	1761	1892	3	[dbg\$match (.input_desc, dbg\$cs_float, 1)
:	1762	1893	3	dbg\$match (.input_desc, dbg\$cs_f_float, 1)] :
:	1763	1894	4	BEGIN
:	1764	1895	4	noun_node [dbg\$l_noun_value] = dsc\$k_dtype_f;
:	1765	1896	4	length_node [dbg\$l_noun_value] = 4;
:	1766	1897	3	END;
:	1767	1898	3	
:	1768	1899	3	[dbg\$match (.input_desc, dbg\$cs_g_float, 1)] :
:	1769	1900	4	BEGIN
:	1770	1901	4	noun_node [dbg\$l_noun_value] = dsc\$k_dtype_g;
:	1771	1902	4	length_node [dbg\$l_noun_value] = 8;
:	1772	1903	3	END;
:	1773	1904	3	
:	1774	1905	3	[dbg\$match (.input_desc, dbg\$cs_h_float, 1)] :
:	1775	1906	4	BEGIN
:	1776	1907	4	noun_node [dbg\$l_noun_value] = dsc\$k_dtype_h;
:	1777	1908	4	length_node [dbg\$l_noun_value] = 16;
:	1778	1909	3	END;
:	1779	1910	3	
:	1780	1911	3	[dbg\$match (.input_desc, dbg\$cs_instruction, 1)] :
:	1781	1912	4	BEGIN
:	1782	1913	4	noun_node [dbg\$l_noun_value] = dsc\$k_dtype_zi;
:	1783	1914	4	length_node [dbg\$l_noun_value] = 0;
:	1784	1915	3	END;
:	1785	1916	3	
:	1786	1917	3	[dbg\$match (.input_desc, dbg\$cs_longword, 1)] :
:	1787	1918	4	BEGIN
:	1788	1919	4	noun_node [dbg\$l_noun_value] = dsc\$k_dtype_l;
:	1789	1920	4	length_node [dbg\$l_noun_value] = 4;
:	1790	1921	3	END;
:	1791	1922	3	
:	1792	1923	3	[dbg\$match (.input_desc, dbg\$cs_octaword, 1)] :
:	1793	1924	4	BEGIN
:	1794	1925	4	noun_node [dbg\$l_noun_value] = dsc\$k_dtype_o;
:	1795	1926	4	length_node [dbg\$l_noun_value] = 16;
:	1796	1927	3	END;
:	1797	1928	3	
:	1798	1929	3	[dbg\$match (.input_desc, dbg\$cs_quadword, 1)] :
:	1799	1930	4	BEGIN
:	1800	1931	4	noun_node [dbg\$l_noun_value] = dsc\$k_dtype_q;
:	1801	1932	4	length_node [dbg\$l_noun_value] = 8;
:	1802	1933	3	END;


```
: 1803      1934      3
: 1804      1935      3
: 1805      1936      4
: 1806      1937      4
: 1807      1938      4
: 1808      1939      3
: 1809      1940      3
: 1810      1941      3
: 1811      1942      4
: 1812      1943      4
: 1813      1944      5
: 1814      1945      5
: 1815      1946      5
: 1816      1947      5
: 1817      1948      4
: 1818      1949      4
: 1819      1950      3
: 1820      1951      3
: 1821      1952      3
: 1822      1953      3
: 1823      1954      2
: 1824      1955      2
: 1825      1956      2
: 1826      1957      2
: 1827      1958      2
: 1828      1959      2
: 1829      1960      3
: 1830      1961      3
: 1831      1962      3
: 1832      1963      2
: 1833      1964      2
: 1834      1965      2
: 1835      1966      2
: 1836      1967      2
: 1837      1968      2
: 1838      1969      3
: 1839      1970      3
: 1840      1971      3
: 1841      1972      2
: 1842      1973      2
: 1843      1974      2
: 1844      1975      2
: 1845      1976      2
: 1846      1977      2
: 1847      1978      3
: 1848      1979      3
: 1849      1980      3
: 1850      1981      3
: 1851      1982      3
: 1852      1983      3
: 1853      1984      3
: 1854      1985      3
: 1855      1986      3
: 1856      1987      3
: 1857      1988      3
: 1858      1989      3
: 1859      1990      2

      [dbg$match (.input_desc, dbg$cs_word, 1)] :
      BEGIN
        noun_node [dbg$l_noun_value] = dsc$k_dtype_w;
        length_node [dbg$l_noun_value] = 2;
      END;

      [OTHERWISE] :
      BEGIN
        .message_vect =
        (IF dbg$match (.input_desc, dbg$cs_cr, 1)
        THEN
          dbg$make_arg_vect (dbg$_needmore)
        ELSE
          dbg$syntax_error (dbg$next_word (.input_desc)));
        RETURN sts$k_severe;
      END;

      TES;

    END;

    ! Parse the SET WATCH command.
    [DBG$MATCH (.INPUT_DESC, DBG$CS_WATCH, 1)]:
    BEGIN
      VERB NODE [DBG$B_VERB_COMPOSITE] = SET WATCH;
      RETURN DBG$EVENT_SYNTAX(.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT);
    END;

    ! Parse the SET WINDOW command.
    [DBG$MATCH(.INPUT_DESC, DBG$CS_WINDOW, 3)]:
    BEGIN
      VERB NODE[DBG$B_VERB_COMPOSITE] = SET WINDOW;
      DBG$SCR_PARSE_SETWIND_CMD(.INPUT_DESC, .VERB_NODE);
    END;

    ! Any other SET command keyword constitutes a syntax error.
    [OTHERWISE]:
    BEGIN
      IF DBG$MATCH (.INPUT_DESC, DBG$CS_CR, 1)
      THEN
        .MESSAGE_VECT = DBG$MAKE_ARG_VECT(DBG$_NEEDMORE)
      ELSE
        .MESSAGE_VECT = DBG$SYNTAX_ERROR(DBG$NEXT_WORD (.INPUT_DESC));
      RETURN STS$K_SEVERE;
    END;

    TES;
```


1991	2
1992	2
1993	2
1994	2
1995	2
1996	1

```
! The SET command has been successfully parsed. Return to the caller.  
RETURN STSSK_SUCCESS;  
END;
```

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

.PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0
```

4C	49	46	5F	45	43	52	53	4E	49	47	52	41	4D	10	00042	P.AAJ:	.ASCII	<16>\MAX_SOURCE_FILES\
							53	4E	49	47	52	41	4D	10	00051			
															00053	P.AAK:	.ASCII	<4>\MODE\
															00058	P.AAL:	.ASCII	<6>\MODULE\
															0005F	P.AAM:	.ASCII	<6>\OUTPUT\
															00066	P.AAN:	.ASCII	<5>\RADIX\
															0006C	P.AAO:	.ASCII	<5>\SCOPE\
															00072	P.AAP:	.ASCII	<6>\SEARCH\
															00079	P.AAQ:	.ASCII	<6>\SOURCE\
															00080	P.AAR:	.ASCII	<4>\STEP\
															00085	P.AAS:	.ASCII	<4>\TASK\
															0008A	P.AAT:	.ASCII	<8>\TERMINAL\
															00093	P.AAU:	.ASCII	<5>\TRACE\
															00099	P.AAV:	.ASCII	<4>\TYPE\
															0009E	P.AAW:	.ASCII	<5>\WATCH\
															000A4	P.AAX:	.ASCII	<6>\WINDOW\
															000AB	P.AAY:	.ASCII	<1>\:\
															000AD	P.AAZ:	.ASCII	<1>\:\
															000AF	P.ABA:	.ASCII	<1>\=\
															000B1	P.ABB:	.ASCII	<1>\/\
															000B3	P.ABC:	.BYTE	1, 13
															000B5	P.ABD:	.BYTE	7
															000B6		.ASCII	\ADDRESS\
															000BD	P.ABE:	.BYTE	7
															000BE		.ASCII	\COMMAND\
															000C5	P.ABF:	.BYTE	9
															000C6		.ASCII	\PROCEDURE\
															000CF	P.ABG:	.BYTE	6
															000D0		.ASCII	\STRING\
															000D6	P.ABH:	.BYTE	5
															000D7		.ASCII	\VALUE\
															000DC	P.ABI:	.BYTE	5
															000DD		.ASCII	\STATE\
															000E2	P.ABJ:	.BYTE	7

4E	41	4F	52	54	52	4F	46	07	000F6	P.ABM:	.ASCII	<7>\FORTRAN\								
		53	53	49	4C	42	05	000FE	P.ABN:	.ASCII	<5>\BLISS\									
		4C	4F	42	4F	43	05	00104	P.ABO:	.ASCII	<5>\COBOL\									
		43	49	53	41	42	05	0010A	P.ABP:	.ASCII	<5>\BASIC\									
				49	4C	50	03	00110	P.ABQ:	.ASCII	<3>\PLI\									
	4C	41	43	53	41	50	06	00114	P.ABR:	.ASCII	<6>\PASCAL\									
						43	01	0011B	P.ABS:	.ASCII	<1>\C\									
				47	50	52	03	0011D	P.ABT:	.ASCII	<3>\RPG\									
				41	44	41	03	00121	P.ABU:	.ASCII	<3>\ADA\									
4E	57	4F	4E	4B	4E	55	07	00125	P.ABV:	.ASCII	<7>\UNKNOWN\									
	59	52	41	4E	49	42	06	0012D	P.ABW:	.ASCII	<6>\BINARY\									
4C	41	4D	49	43	45	44	07	00134	P.ABX:	.ASCII	<7>\DECIMAL\									
54	4C	55	41	46	45	44	07	0013C	P.ABY:	.ASCII	<7>\DEFAULT\									
54	41	4F	4C	46	5F	47	07	00144	P.ABZ:	.ASCII	<7>\G_FLOAT\									
4C	41	4D	49	43	45	44	07	0014C	P.ACA:	.ASCII	<11>\HEXADECIMAL\									
				44	41	50	06	00158	P.ACB:	.ASCII	<6>\KEYPAD\									
	54	41	4F	4C	46	5F	47	4F	4E	09	0015F	P.ACC:	.ASCII	<9>\NOG_FLOAT\						
		44	41	50	59	45	4B	4F	4E	08	00169	P.ACD:	.ASCII	<8>\NOKEYPAD\						
		4E	45	45	52	43	53	4F	4E	08	00172	P.ACE:	.ASCII	<8>\NOSCREEN\						
43	49	4C	4F	42	4D	59	53	4F	4E	0A	0017B	P.ACF:	.ASCII	<10>\NOSYMBOLIC\						
				4C	41	54	43	4F	05	00186	P.ACG:	.ASCII	<5>\OCTAL\							
				4E	45	45	52	43	53	06	0018C	P.ACH:	.ASCII	<6>\SCREEN\						
	43	49	4C	4F	42	4D	59	53	08	00193	P.ACI:	.ASCII	<8>\SYMBOLIC\							
									03	0019C	P.ACJ:	.BYTE	3							
						4C	4C	41	0019D		.ASCII	\ALL\								
								08	001A0	P.ACK:	.BYTE	8								
	45	54	41	43	4F	4C	4C	41	001A1		.ASCII	\ALLOCATE\								
								06	001A9	P.ACL:	.BYTE	6								
		54	49	41	57	4F	4E	001AA		.ASCII	\NOWAIT\									
4E	24	47	42	44	5C	57	4F	48	53	4E	47	42	44	1A	001B0	P.ACM:	.BYTE	1, 44		
		30	31	20	54	45	53	5F	45	53	52	41	50	001B2	P.ACN:	.ASCII	<26>\DBGNSHOW\<92>\DBG\$NPARSE_SET 10\			
										47	4F	4C	03	001CD	P.ACO:	.ASCII	<3>\LOG\			
										45	45	52	43	53	0A	001D1	P.ACP:	.ASCII	<5>\NOLOG\	
										45	52	43	53	4F	4E	0C	001D7	P.ACQ:	.ASCII	<10>\SCREEN_LOG\
47	4F	4C	5F	4C	41	4E	49	4D	52	45	54	4F	4E	0A	001E2	P.ACR:	.ASCII	<12>\NOSCREEN_LOG\		
		4C	41	4E	49	4D	52	45	54	4F	4E	0A	001F8	P.ACS:	.ASCII	<8>\TERMINAL\				
						59	46	49	52	45	56	06	00203	P.ACT:	.ASCII	<10>\NOTERMINAL\				
				59	46	49	52	45	56	4F	4E	08	0020A	P.ACU:	.ASCII	<6>\VERIFY\				
						59	52	41	4E	49	42	06	00213</							

52	45	49	46	49	54	4E	45	44	0A	00262	P.ADH:	.BYTE	10				
									49	00263		.ASCII	\IDENTIFIER\				
									06	0026D	P.ADI:	.BYTE	6				
				45	4C	55	44	4F	4D	0026E		.ASCII	\MODULE\				
									02	00274	P.ADJ:	.BYTE	2				
								43	41	00275		.ASCII	\AC\				
									02	00277	P.ADK:	.BYTE	2				
								44	41	00278		.ASCII	\AD\				
									05	0027A	P.ADL:	.BYTE	5				
				43	49	43	53		41	0027B		.ASCII	\ASCIC\				
									05	00280	P.ADM:	.BYTE	5				
				44	49	43	53		41	00281		.ASCII	\ASCID\				
									05	00286	P.ADN:	.BYTE	5				
				49	49	43	53		41	00287		.ASCII	\ASCII\				
									05	0028C	P.ADO:	.BYTE	5				
				5A	49	43	53		41	0028D		.ASCII	\ASCIZ\				
									02	00292	P.ADP:	.BYTE	2				
								5A	41	00293		.ASCII	\AZ\				
									04	00295	P.ADQ:	.BYTE	4				
					45	54	59		42	00296		.ASCII	\BYTE\				
								3A	01	0029A	P.ADR:	.BYTE	1, 58				
									07	0029C	P.ADS:	.BYTE	7				
				54	41	4F	4C	46	5F	44	0029D		.ASCII	\D_FLOAT\			
									05	002A4	P.ADT:	.BYTE	5				
					54	41	4F	4C	46	002A5		.ASCII	\FLOAT\				
									07	002AA	P.ADU:	.BYTE	7				
				54	41	4F	4C	46	5F	46	002AB		.ASCII	\F_FLOAT\			
									07	002B2	P.ADV:	.BYTE	7				
				54	41	4F	4C	46	5F	47	002B3		.ASCII	\G_FLOAT\			
									07	002BA	P.ADE:	.BYTE	7				
				54	41	4F	4C	46	5F	48	002BB		.ASCII	\H_FLOAT\			
									06	002C2	P.ADX:	.BYTE	6				
				44	45	4B	43	41	50	002C3		.ASCII	\PACKED\				
									0B	002C9	P.ADY:	.BYTE	11				
4E	4F	49	54	43	55	52	54	53	4E	49	002CA		.ASCII	\INSTRUCTION\			
									09	002D5	P.ADZ:	.BYTE	9				
				45	4D	49	54	5F	45	54	41	44	002D6		.ASCII	\DATE_TIME\	
									08	002DF	P.AEA:	.BYTE	8				
				44	52	4F	57	47	4E	4F	4C	002E0		.ASCII	\LONGWORD\		
									08	002E8	P.AEB:	.BYTE	8				
				44	52	4F	57	41	54	43	4F	002E9		.ASCII	\OCTAWORD\		
									08	002F1	P.AEC:	.BYTE	8				
				45	44	49	52	52	45	56	4F	002F2		.ASCII	\OVERRIDE\		
									08	002FA	P.AED:	.BYTE	8				
				44	52	4F	57	44	41	55	51	002FB		.ASCII	\QUADWORD\		
									04	00303	P.AEE:	.BYTE	4				
								44	52	4F	57	00304		.ASCII	\WORD\		
									30	00308	P.AEF:	.ASCII	\0\				

.PSECT DBG\$OWN,NOEXE, PIC,2

00000 ZERO_DESC:

.BLKB 12

.PSECT DBG\$GLOBAL,NOEXE, PIC,2

00000000 00000 DBG\$GL_DEVELOPER::

.LONG 0

```

DBG$CS_BREAK= P.AAA
DBG$CS_DEFINE= P.AAB
DBG$CS_DEVELOPER= P.AAC
DBG$CS_DISPLAY= P.AAD
DBG$CS_EXCEPTION= P.AAE
DBG$CS_LANGUAGE= P.AAF
DBG$CS_LOG= P.AAG
DBG$CS_KEY= P.AAH
DBG$CS_MARGINS= P.AAI
DBG$CS_MAX_SOURCE_FILES= P.AAJ
DBG$CS_MODE= P.AAK
DBG$CS_MODULE= P.AAL
DBG$CS_OUTPUT= P.AAM
DBG$CS_RADIX= P.AAN
DBG$CS_SCOPE= P.AAO
DBG$CS_SEARCH= P.AAP
DBG$CS_SOURCE= P.AAQ
DBG$CS_STEP= P.AAR
DBG$CS_TASK= P.AAS
DBG$CS_TERMINAL= P.AAT
DBG$CS_TRACE= P.AAU
DBG$CS_TYPE= P.AAV
DBG$CS_WATCH= P.AAW
DBG$CS_WINDOW= P.AAX
DBG$CS_COLON= P.AAY
DBG$CS_COMMA= P.AAZ
DBG$CS_EQUAL= P.ABA
DBG$CS_SLASH= P.ABB
DBG$CS_CR= P.ABC
DBG$CS_ADDRESS= P.ABD
DBG$CS_COMMAND= P.ABE
DBG$CS_PROCEDURE= P.ABF
DBG$CS_STRING= P.ABG
DBG$CS_VALUE= P.ABH
DBG$CS_STATE= P.ABI
DBG$CS_NOSTATE= P.ABJ
DBG$CS_NOLOG= P.ABK
DBG$CS_MACRO= P.ABL
DBG$CS_FORTRAN= P.ABM
DBG$CS_BLISS= P.ABN
DBG$CS_COBOL= P.ABO
DBG$CS_BASIC= P.ABP
DBG$CS_PLI= P.ABQ
DBG$CS_PASCAL= P.ABR
DBG$CS_C= P.ABS
DBG$CS_RPG= P.ABT
DBG$CS_ADA= P.ABU
DBG$CS_UNKNOWN= P.ABV
DBG$CS_BINARY= P.ABW
DBG$CS_DECIMAL= P.ABX
DBG$CS_DEFAULT= P.ABY
DBG$CS_G_FLOAT= P.ABZ
DBG$CS_HEXADECIMAL= P.ACA
DBG$CS_KEYPAD= P.ACB

```



```

DBG$CS_NOG_FLOAT= P.ACC
DBG$CS_NOKEYPAD= P.ACD
DBG$CS_NOSCREEN= P.ACE
DBG$CS_NOSYMBOLS= P.ACF
DBG$CS_OCTAL= P.ACG
DBG$CS_SCREEN= P.ACH
DBG$CS_SYMBOLS= P.ACI
DBG$CS_ALL= P.ACJ
DBG$CS_ALLOCATE= P.ACK
DBG$CS_NOWAIT= P.ACL
DBG$CS_COMMA= P.ACM
DBG$CS_LOG= P.ACO
DBG$CS_NOLOG= P.ACP
DBG$CS_SCREEN_LOG= P.ACQ
DBG$CS_NOSCREEN_LOG= P.ACR
DBG$CS_TERMINAL= P.ACS
DBG$CS_NOTERMINAL= P.ACT
DBG$CS_VERIFY= P.ACU
DBG$CS_NOVERIFY= P.ACV
DBG$CS_BINARY= P.ACW
DBG$CS_DECIMAL= P.ACX
DBG$CS_DEFAULT= P.ACY
DBG$CS_HEXADECEIMAL= P.ACZ
DBG$CS_INPUT= P.ADA
DBG$CS_OCTAL= P.ADB
DBG$CS_OUTPUT= P.ADC
DBG$CS_OVERRIDE= P.ADD
DBG$CS_ALL= P.ADE
DBG$CS_NEXT= P.ADF
DBG$CS_STRING= P.ADG
DBG$CS_IDENT= P.ADH
DBG$CS_MODULE= P.ADI
DBG$CS_AC= P.ADJ
DBG$CS_AD= P.ADK
DBG$CS_ASCIC= P.ADL
DBG$CS_ASCID= P.ADM
DBG$CS_ASCII= P.ADN
DBG$CS_ASCIZ= P.ADO
DBG$CS_AZ= P.ADP
DBG$CS_BYTE= P.ADQ
DBG$CS_COLON= P.ADR
DBG$CS_D_FLOAT= P.ADS
DBG$CS_FLOAT= P.ADT
DBG$CS_F_FLOAT= P.ADU
DBG$CS_G_FLOAT= P.ADV
DBG$CS_H_FLOAT= P.ADW
DBG$CS_PACKED= P.ADX
DBG$CS_INSTRUCTION= P.ADY
DBG$CS_DATE_TIME= P.ADZ
DBG$CS_LONGWORD= P.AEA
DBG$CS_OCTAWORD= P.AEB
DBG$CS_OVERRIDE= P.AEC
DBG$CS_QUADWORD= P.AED
DBG$CS_WORD= P.AEE
      .EXTRN DBG$EVENT_SYNTAX
      .EXTRN DBG$EVENT_SEMANTICS
      .EXTRN DBG$NGET_LENGTH

```



```
.EXTRN DBG$NGET_DIR_LIST
.EXTRN DBG$NGET_TRANS_RADIX
.EXTRN DBG$SCR_EXECUTE_DISPLAY_CMD
.EXTRN DBG$SCR_EXECUTE_SETTERM_CMD
.EXTRN DBG$SCR_EXECUTE_SETWIND_CMD
.EXTRN DBG$SCR_PARSE_DISPLAY_CMD
.EXTRN DBG$SCR_PARSE_SETTERM_CMD
.EXTRN DBG$SCR_PARSE_SETWIND_CMD
.EXTRN DBG$SCR_SCREEN_MODE
.EXTRN DBG$STA_GETSourcemod
.EXTRN DBG$SRC_SET_SOURCE
.EXTRN DBG$NSAVE_DECIMAL_INTEGER
.EXTRN DBG$NSAVE_INTEGER
.EXTRN DBG$NSAVE_STRING
.EXTRN DBG$NGET_TYPE, DBG$PRINT
.EXTRN DBG$NEWLINE, DBG$SET_SEARCH_LVL
.EXTRN DBG$SET_STP_LVL
.EXTRN DBG$SET_DEFINE_LVL
.EXTRN DBG$SET_MOD_LVL
.EXTRN DBG$NPARSE_SET_TASK
.EXTRN DBG$NEXECUTE_SET_TASK
.EXTRN DBG$RST_SETSCOPE
.EXTRN DBG$NPARSE_SCOPE_LIST
.EXTRN DBG$NSHOW_MARGINS
.EXTRN DBG$SRC_SET_MAX_FILES
.EXTRN DBG$NPATHDESC TO CS
.EXTRN DBG$RST_SETMOD, DBG$GET_MEMORY
.EXTRN DBG$REL_MEMORY, DBG$GET_TEMPMEM
.EXTRN DBG$NSAVE_FILESP
.EXTRN DBG$NMATCH, DBG$NPARSE_ADDRESS
.EXTRN DBG$NNEXT_WORD, DBG$NOOT_INFO
.EXTRN DBG$NMAKE_ARG_VECT
.EXTRN DBG$NSYNTAX_ERROR
.EXTRN DBG$SET_LANG, DBG$READ_KEY_INFO
.EXTRN SMG$SET_DEFAULT_STATE
.EXTRN SMG$SET_KEYPAD_MODE
.EXTRN DBG$GB_KEYPAD_INPUT
.EXTRN DBG$GL_KEY_TABLE_ID
.EXTRN DBG$GL_KEYBOARD_ID
.EXTRN DBG$GB_RADIX, DBG$GL_GBLTYP
.EXTRN DBG$GW_GBLLNGTH
.EXTRN DBG$GL_DFLTYP, DBG$GW_DFLTLENG
.EXTRN DBG$GB_LANGUAGE
.EXTRN DBG$GB_SEARCH_PTR
.EXTRN DBG$GB_MOD_PTR, DBG$GB_DEFINE_PTR
.EXTRN DBG$RUNFRAME, DBG$GB_RESIGNAL
.EXTRN DBG$GL_LOG_BUF, DBG$GL_CONTEXT
.EXTRN DBG$GB_DEF_OUT, DBG$GL_LOGFAB
.EXTRN DBG$GL_LOGRAB, DBG$GL_COGNAM
.EXTRN DBG$GB_LOGFSR, DBG$GB_LOGFSE
.EXTRN DBG$GL_SCREEN_LOG
.EXTRN DBG$GB_STP_PTR, DBG$SRC_LEFT_MARGIN
.EXTRN DBG$SRC_RIGHT_MARGIN
.EXTRN DBG$GL_ORIG_COMMAND_PTR
.EXTRN DBG$GL_UPCASE_COMMAND_PTR

.PSECT DBG$CODE, NOWRT, SHR, PIC, 0
```


			OFFC	00000	.ENTRY	DBG\$NPARSE_SET, Save R2,R3,R4,R5,R6,R7,R8,-	
	5E		18	C2 00002	SUBL2	R9,R10,R11	0452
			04	DD 00005	PUSHL	#24, SP	
00000000G	00		01	FB 00007	CALLS	#1, DBG\$GET_TEMP MEM	0543
	58		50	DD 0000E	MOVL	R0, NOUN_NODE	
	59	08	AC	DD 00011	MOVL	VERB_NODE, R9	0548
08	A9		58	DD 00015	MOVL	NOUN_NODE, 8(R9)	
			01	DD 00019	PUSHL	#1	0559
		00000000'	EF	9F 0001B	PUSHAB	DBG\$CS_BREAK	
	57	04	AC	DD 00021	MOVL	INPUT_DESC, R7	
			57	DD 00025	PUSHL	R7	
00000000G	00		03	FB 00027	CALLS	#3, DBG\$NMATCH	
	01		50	D1 0002E	CMPL	R0, #1	
			07	12 00031	BNEQ	1\$	
01	A9		01	90 00033	MOVB	#1, 1(R9)	0561
			1054	31 00037	BRW	205\$	0564
			01	DD 0003A	PUSHL	#1	0569
		00000000'	EF	9F 0003C	PUSHAB	DBG\$CS_DEFINE	
			57	DD 00042	PUSHL	R7	
00000000G	00		03	FB 00044	CALLS	#3, DBG\$NMATCH	
	01		50	D1 0004B	CMPL	R0, #1	
			52	12 0004E	BNEQ	5\$	
01	A9		17	90 00050	MOVB	#23, 1(R9)	0581
			01	DD 00054	PUSHL	#1	0590
		00000000'	EF	9F 00056	PUSHAB	DBG\$CS_ADDRESS	
			57	DD 0005C	PUSHL	R7	
00000000G	00		03	FB 0005E	CALLS	#3, DBG\$NMATCH	
	01		50	D1 00065	CMPL	R0, #1	
			03	12 00068	BNEQ	2\$	
		0A17	31	0006A	BRW	130\$	
			01	DD 0006D	PUSHL	#1	0593
		00000000'	EF	9F 0006F	PUSHAB	DBG\$CS_COMMAND	
			57	DD 00075	PUSHL	R7	
00000000G	00		03	FB 00077	CALLS	#3, DBG\$NMATCH	
	01		50	D1 0007E	CMPL	R0, #1	
			03	12 00081	BNEQ	3\$	
		0A6D	31	00083	BRW	138\$	
			01	DD 00086	PUSHL	#1	0602
		00000000'	EF	9F 00088	PUSHAB	DBG\$CS_VALUE	
			57	DD 0008E	PUSHL	R7	
00000000G	00		03	FB 00090	CALLS	#3, DBG\$NMATCH	
	01		50	D1 00097	CMPL	R0, #1	
			03	13 0009A	BEQL	4\$	
		0D0E	31	0009C	BRW	169\$	
		02E9	31	0009F	BRW	37\$	
			09	DD 000A2	PUSHL	#9	0620
		00000000'	EF	9F 000A4	PUSHAB	DBG\$CS_DEVELOPER	
			57	DD 000AA	PUSHL	R7	
00000000G	00		03	FB 000AC	CALLS	#3, DBG\$NMATCH	
	01		50	D1 000B3	CMPL	R0, #1	
			03	13 000B6	BEQL	6\$	
		0084	31	000B8	BRW	13\$	
01	A9		16	90 000BB	MOVB	#22, 1(R9)	0625
	52	08	A9	9E 000BF	MOVAB	8(R9), LINK	0626
			01	DD 000C3	PUSHL	#1	0627

		00000000'	EF 9F 000C5	PUSHAB	DBG\$CS_CR	
			57 DD 000CB	PUSHL	R7	
00000000G	00		03 FB 000CD	CALLS	#3, DBG\$NMATCH	
	03		50 E9 000D4	BLBC	R0, 7\$	
		OCEA	31 000D7	BRW	170\$	
	7E		57 7D 000DA 7\$:	MOVQ	R7, -(SP)	0632
00000000G	00		02 FB 000DD	CALLS	#2, DBG\$NSAVE_INTEGER	
	03		50 E8 000E4	BLBS	R0, 8\$	
		OFF3	31 000E7	BRW	212\$	
			68 D5 000EA 8\$:	TSTL	(NOUN_NODE)	0636
			05 19 000EC	BLSS	9\$	
	1F		68 D1 000EE	CMPL	(NOUN_NODE), #31	0637
			09 15 000F1	BLEQ	10\$	
		00028248	8F DD 000F3 9\$:	PUSHL	#164424	0640
		OCCE	31 000F9	BRW	171\$	
	52	08	A8 9E 000FC 10\$:	MOVAB	8(R8), LINK	0644
			01 DD 00100	PUSHL	#1	0645
		00000000'	EF 9F 00102	PUSHAB	DBG\$CS_COMMA	
			57 DD 00108	PUSHL	R7	
00000000G	00		03 FB 0010A	CALLS	#3, DBG\$NMATCH	
	1A		50 E8 00111	BLBS	R0, 12\$	
			01 DD 00114	PUSHL	#1	0648
		00000000'	EF 9F 00116	PUSHAB	DBG\$CS_CR	
			57 DD 0011C	PUSHL	R7	
00000000G	00		03 FB 0011E	CALLS	#3, DBG\$NMATCH	
	03		50 E8 00125	BLBS	R0, 11\$	
		OF9C	31 00128	BRW	210\$	
		OEC9	31 0012B 11\$:	BRW	195\$	
			04 DD 0012E 12\$:	PUSHL	#4	0660
00000000G	00		01 FB 00130	CALLS	#1, DBG\$GET_TEMPMEM	
	58		50 D0 00137	MOVL	R0, NOUN_NODE	
	62		58 D0 0013A	MOVL	NOUN_NODE, (LINK)	0661
			9B 11 0013D	BRB	7\$	0630
			03 DD 0013F 13\$:	PUSHL	#3	0679
		00000000'	EF 9F 00141	PUSHAB	DBG\$CS_DISPLAY	
			57 DD 00147	PUSHL	R7	
00000000G	00		03 FB 00149	CALLS	#3, DBG\$NMATCH	
	01		50 D1 00150	CMPL	R0, #1	
			13 12 00153	BNEQ	14\$	
	01	A9	18 90 00155	MOVB	#24, 1(R9)	0681
			59 DD 00159	PUSHL	R9	0682
			01 DD 0015B	PUSHL	#1	
			57 DD 0015D	PUSHL	R7	
00000000G	00		03 FB 0015F	CALLS	#3, DBG\$SCR_PARSE_DISPLAY_CMD	
			34 11 00166	BRB	16\$	0553
			01 DD 00168 14\$:	PUSHL	#1	0688
		00000000'	EF 9F 0016A	PUSHAB	DBG\$CS_EXCEPTION	
			57 DD 00170	PUSHL	R7	
00000000G	00		03 FB 00172	CALLS	#3, DBG\$NMATCH	
	01		50 D1 00179	CMPL	R0, #1	
			21 12 0017C	BNEQ	17\$	
			01 DD 0017E	PUSHL	#1	0693
		00000000'	EF 9F 00180	PUSHAB	DBG\$CS_BREAK	
			57 DD 00186	PUSHL	R7	
00000000G	00		03 FB 00188	CALLS	#3, DBG\$NMATCH	
	03		50 E8 0018F	BLBS	R0, 15\$	
		OC18	31 00192	BRW	169\$	

01	A9	03	90	00195	15\$:	MOVB	#3, 1(R9)	0707	
		04	A9	7C	00199	CLRL	4(R9)	0713	
		0F	42	31	0019C	BRW	213\$	0553	
			01	DD	0019F	17\$:	PUSHL	#1	0719
		00000000'	EF	9F	001A1	PUSHAB	DBG\$CS_KEY		
			57	DD	001A7	PUSHL	R7		
00000000G	00		03	FB	001A9	CALLS	#3, DBG\$NMATCH		
	01		50	D1	001B0	CMPL	R0, #1		
			03	13	001B3	BEQL	18\$		
		01	22	31	001B5	BRW	30\$		
	0D	00000000G	00	E8	001B8	18\$:	BLBS	DBG\$GB_KEYPAD_INPUT, 19\$	0735
		00028D18	8F	DD	001BF	PUSHL	#167192	0737	
00000000G	00		01	FB	001C5	CALLS	#1, LIB\$SIGNAL		
	01		1D	90	001CC	19\$:	MOVB	#29, 1(R9)	0739
			02	DD	001D0	PUSHL	#2	0744	
00000000G	00		01	FB	001D2	CALLS	#1, DBG\$GET_TEMP_MEM		
	52		50	D0	001D9	MOVL	R0, TEMP_KEY_DESC		
	62	020E0000	8F	D0	001DC	MOVL	#34471936, (TEMP_KEY_DESC)	0745	
		04	A2	D4	001E3	CLRL	4(TEMP_KEY_DESC)	0748	
			52	DD	001E6	PUSHL	TEMP_KEY_DESC	0750	
			7E	D4	001E8	CLRL	-(SP)		
		00000000G	00	9F	001EA	PUSHAB	DBG\$GL_KEY_TABLE_ID		
00000000G	00		03	FB	001F0	CALLS	#3, SMG\$SET_DEFAULT_STATE		
	04		01	D0	001F7	MOVL	#1, 4(R9)	0754	
			52	D0	001FB	20\$:	MOVL	TEMP_KEY_DESC, (NOUN_NODE)	0756
			01	DD	001FE	21\$:	PUSHL	#1	0758
		00000000'	EF	9F	00200	PUSHAB	DBG\$CS_CR		
			57	DD	00206	PUSHL	R7		
00000000G	00		03	FB	00208	CALLS	#3, DBG\$NMATCH		
	8A		50	E8	0020F	BLBS	R0, 16\$		
			67	B5	00212	TSTW	(R7)	0759	
			86	13	00214	BEQL	16\$		
		00000000'	01	DD	00216	PUSHL	#1	0761	
			EF	9F	00218	PUSHAB	DBG\$CS_SLASH		
			57	DD	0021E	PUSHL	R7		
00000000G	00		03	FB	00220	CALLS	#3, DBG\$NMATCH		
	03		50	E8	00227	BLBS	R0, 22\$		
		0E	9A	31	0022A	BRW	210\$		
			03	DD	0022D	22\$:	PUSHL	#3	0766
		00000000'	EF	9F	0022F	PUSHAB	DBG\$CS_NOLOG		
			57	DD	00235	PUSHL	R7		
00000000G	00		03	FB	00237	CALLS	#3, DBG\$NMATCH		
	01		50	D1	0023E	CMPL	R0, #1		
			05	12	00241	BNEQ	24\$		
		04	A9	D4	00243	CLRL	4(R9)	0768	
			B6	11	00246	23\$:	BRB	21\$	0763
			03	DD	00248	24\$:	PUSHL	#3	0771
		00000000'	EF	9F	0024A	PUSHAB	DBG\$CS_NOSTATE		
			57	DD	00250	PUSHL	R7		
00000000G	00		03	FB	00252	CALLS	#3, DBG\$NMATCH		
	01		50	D1	00259	CMPL	R0, #1		
			13	12	0025C	BNEQ	25\$		
			52	DD	0025E	PUSHL	TEMP_KEY_DESC	0776	
			7E	D4	00260	CLRL	-(SP)		
		00000000G	00	9F	00262	PUSHAB	DBG\$GL_KEY_TABLE_ID		
00000000G	00		03	FB	00268	CALLS	#3, SMG\$SET_DEFAULT_STATE		
			8A	11	0026F	BRB	20\$	0777	

		01 DD 00271 25\$:	PUSHL #1	0780
		EF 9F 00273	PUSHAB DBG\$CS_LOG	
00000000G	00	57 DD 00279	PUSHL R7	
	01	03 FB 0027B	CALLS #3, DBG\$NMATCH	
		50 D1 00282	CMPL R0, #1	
		06 12 00285	BNEQ 26\$	
04	A9	01 D0 00287	MOVL #1, 4(R9)	0782
		B9 11 0028B	BRB 23\$	0763
		01 DD 0028D 26\$:	PUSHL #1	0785
		EF 9F 0028F	PUSHAB DBG\$CS_STATE	
00000000G	00	57 DD 00295	PUSHL R7	
	01	03 FB 00297	CALLS #3, DBG\$NMATCH	
		50 D1 0029E	CMPL R0, #1	
		03 13 002A1	BEQL 28\$	
		0B07 31 002A3 27\$:	BRW 169\$	
		01 DD 002A6 28\$:	PUSHL #1	0789
		EF 9F 002A8	PUSHAB DBG\$CS_EQUAL	
00000000G	00	57 DD 002AE	PUSHL R7	
	E9	03 FB 002B0	CALLS #3, DBG\$NMATCH	
		50 E9 002B7	BLBC R0, 27\$	
	04	A2 D4 002BA	CLRL 4(TEMP_KEY_DESC)	0800
		62 B4 002BD	CLRW (TEMP_KEY_DESC)	0801
	0C	AC DD 002BF	PUSHL MESSAGE_VECT	0804
		52 DD 002C2	PUSHL TEMP_KEY_DESC	0803
		57 DD 002C4	PUSHL R7	0802
00000000G	00	03 FB 002C6	CALLS #3, DBG\$READ_KEY_INFO	
	53	50 D0 002CD	MOVL R0, STATUS	
	03	53 E9 002D0	BLBC STATUS, 29\$	0805
		FF28 31 002D3	BRW 21\$	
	50	53 D0 002D6 29\$:	MOVL STATUS, R0	0807
		04 002D9	RET	
		02 DD 002DA 30\$:	PUSHL #2	0828
		EF 9F 002DC	PUSHAB DBG\$CS_LANGUAGE	
00000000G	00	57 DD 002E2	PUSHL R7	
	01	03 FB 002E4	CALLS #3, DBG\$NMATCH	
		50 D1 002EB	CMPL R0, #1	
		03 13 002EE	BEQL 31\$	
		0131 31 002F0	BRW 49\$	
01	A9	04 90 002F3 31\$:	MOVB #4, 1(R9)	0845
		02 DD 002F7	PUSHL #2	0850
		EF 9F 002F9	PUSHAB DBG\$CS_MACRO	
00000000G	00	57 DD 002FF	PUSHL R7	
	01	03 FB 00301	CALLS #3, DBG\$NMATCH	
		50 D1 00308	CMPL R0, #1	
		04 12 0030B	BNEQ 32\$	
		68 D4 0030D	CLRL (NOUN_NODE)	0851
		7D 11 0030F	BRB 38\$	
		02 DD 00311 32\$:	PUSHL #2	0853
		EF 9F 00313	PUSHAB DBG\$CS_FORTRAN	
00000000G	00	57 DD 00319	PUSHL R7	
	01	03 FB 0031B	CALLS #3, DBG\$NMATCH	
		50 D1 00322	CMPL R0, #1	
		03 12 00325	BNEQ 33\$	
		075A 31 00327	BRW 130\$	
		02 DD 0032A 33\$:	PUSHL #2	0856
		EF 9F 0032C	PUSHAB DBG\$CS_BLISS	
		57 DD 00332	PUSHL R7	

00000000G	00	03	FB	00334	CALLS	#3, DBG\$NMATCH	
	01	50	D1	0033B	CMPL	R0, #1	
		03	12	0033E	BNEQ	34\$	
		07B0	31	00340	BRW	138\$	
		02	DD	00343	PUSHL	#2	0859
	00000000'	EF	9F	00345	PUSHAB	DBG\$CS_COBOL	
		57	DD	0034B	PUSHL	R7	
00000000G	00	03	FB	0034D	CALLS	#3, DBG\$NMATCH	
	01	50	D1	00354	CMPL	R0, #1	
		03	12	00357	BNEQ	35\$	
		075E	31	00359	BRW	134\$	
		02	DD	0035C	PUSHL	#2	0862
	00000000'	EF	9F	0035E	PUSHAB	DBG\$CS_BASIC	
		57	DD	00364	PUSHL	R7	
00000000G	00	03	FB	00366	CALLS	#3, DBG\$NMATCH	
	01	50	D1	0036D	CMPL	R0, #1	
		03	12	00370	BNEQ	36\$	
		0760	31	00372	BRW	136\$	
		02	DD	00375	PUSHL	#2	0865
	00000000'	EF	9F	00377	PUSHAB	DBG\$CS_PLI	
		57	DD	0037D	PUSHL	R7	
00000000G	00	03	FB	0037F	CALLS	#3, DBG\$NMATCH	
	01	50	D1	00386	CMPL	R0, #1	
		05	12	00389	BNEQ	39\$	
	68	05	D0	0038B	MOVL	#5, (NOUN_NODE)	0866
		73	11	0038E	BRB	45\$	
		02	DD	00390	PUSHL	#2	0868
	00000000'	EF	9F	00392	PUSHAB	DBG\$CS_PASCAL	
		57	DD	00398	PUSHL	R7	
00000000G	00	03	FB	0039A	CALLS	#3, DBG\$NMATCH	
	01	50	D1	003A1	CMPL	R0, #1	
		05	12	003A4	BNEQ	40\$	
	68	06	D0	003A6	MOVL	#6, (NOUN_NODE)	0869
		76	11	003A9	BRB	48\$	
		01	DD	003AB	PUSHL	#1	0871
	00000000'	EF	9F	003AD	PUSHAB	DBG\$CS_C	
		57	DD	003B3	PUSHL	R7	
00000000G	00	03	FB	003B5	CALLS	#3, DBG\$NMATCH	
	01	50	D1	003BC	CMPL	R0, #1	
		03	12	003BF	BNEQ	41\$	
		06DB	31	003C1	BRW	132\$	
		02	DD	003C4	PUSHL	#2	0874
	00000000'	EF	9F	003C6	PUSHAB	DBG\$CS_RPG	
		57	DD	003CC	PUSHL	R7	
00000000G	00	03	FB	003CE	CALLS	#3, DBG\$NMATCH	
	01	50	D1	003D5	CMPL	R0, #1	
		05	12	003D8	BNEQ	42\$	
	68	08	D0	003DA	MOVL	#8, (NOUN_NODE)	0875
		42	11	003DD	BRB	48\$	
	04	00000000'	EF	E8	003DF	BLBS	DBG\$GL_DEVELOPER, 43\$
		50	D4	003E6	CLRL	R0	0877
		11	11	003E8	BRB	44\$	
		02	DD	003EA	PUSHL	#2	0878
	00000000'	EF	9F	003EC	PUSHAB	DBG\$CS_ADA	
		57	DD	003F2	PUSHL	R7	
00000000G	00	03	FB	003F4	CALLS	#3, DBG\$NMATCH	
	01	50	D1	003FB	CMPL	R0, #1	0877

		05	12	003FE	BNEQ	46\$		
68		09	DD	00400	MOVL	#9, (NOUN_NODE)		0879
		1C	11	00403	BRB	48\$		
		03	DD	00405	PUSHL	#3		0881
	00000000'	EF	9F	00407	PUSHAB	DBG\$CS_UNKNOWN		
		57	DD	0040D	PUSHL	R7		
00000000G	00	03	FB	0040F	CALLS	#3, DBG\$NMATCH		
	01	50	D1	00416	CMPL	R0, #1		
		03	13	00419	BEQL	47\$		
		098F	31	0041B	BRW	169\$		
68		0A	DD	0041E	MOVL	#10, (NOUN_NODE)		0882
		0CBD	31	00421	BRW	213\$		
		02	DD	00424	PUSHL	#2		0901
	00000000'	EF	9F	00426	PUSHAB	DBG\$CS_LOG		
		57	DD	0042C	PUSHL	R7		
00000000G	00	03	FB	0042E	CALLS	#3, DBG\$NMATCH		
	01	50	D1	00435	CMPL	R0, #1		
		17	12	00438	BNEQ	52\$		
01	A9	05	90	0043A	MOVB	#5, 1(R9)		0903
		AC	DD	0043E	PUSHL	MESSAGE_VECT		0906
	7E	57	7D	00441	MOVQ	R7, -(SP)		0905
00000000G	00	03	FB	00444	CALLS	#3, DBG\$NSAVE_FILESP		
	D3	50	E8	0044B	BLBS	R0, 48\$		
		0C8C	31	0044E	BRW	212\$		0908
		03	DD	00451	PUSHL	#3		0911
	00000000'	EF	9F	00453	PUSHAB	DBG\$CS_MARGINS		
		57	DD	00459	PUSHL	R7		
00000000G	00	03	FB	0045B	CALLS	#3, DBG\$NMATCH		
	01	50	D1	00462	CMPL	R0, #1		
		03	13	00465	BEQL	53\$		
		00B3	31	00467	BRW	60\$		
01	A9	14	90	0046A	MOVB	#20, 1(R9)		0917
		01	DD	0046E	PUSHL	#1		0926
	00000000'	EF	9F	00470	PUSHAB	DBG\$CS_COLON		
		57	DD	00476	PUSHL	R7		
00000000G	00	03	FB	00478	CALLS	#3, DBG\$NMATCH		
	1B	50	E9	0047F	BLBC	R0, 54\$		
		0C	AC	DD	00482	PUSHL	MESSAGE_VECT	0934
		04	AE	9F	00485	PUSHAB	RIGHT_MARGIN	0933
		57	DD	00488	PUSHL	R7		
00000000G	00	03	FB	0048A	CALLS	#3, DBG\$NSAVE_DECIMAL_INTEGER		
	BA	50	E9	00491	BLBC	R0, 51\$		
	52	00	DD	00494	MOVL	DBG\$SRC_LEFT_MARGIN, LEFT_MARGIN		0937
		5E	11	0049B	BRB	58\$		0926
		0C	AC	DD	0049D	PUSHL	MESSAGE_VECT	0943
		04	AE	9F	004A0	PUSHAB	RIGHT_MARGIN	0942
		57	DD	004A3	PUSHL	R7		
00000000G	00	03	FB	004A5	CALLS	#3, DBG\$NSAVE_DECIMAL_INTEGER		
	9F	50	E9	004AC	BLBC	R0, 51\$		
		01	DD	004AF	PUSHL	#1		0950
	00000000'	EF	9F	004B1	PUSHAB	DBG\$CS_COLON		
		57	DD	004B7	PUSHL	R7		
00000000G	00	03	FB	004B9	CALLS	#3, DBG\$NMATCH		
	35	50	E9	004C0	BLBC	R0, 57\$		
	52	6E	DD	004C3	MOVL	RIGHT_MARGIN, LEFT_MARGIN		0957
		01	DD	004C6	PUSHL	#1		0958
	00000000'	EF	9F	004C8	PUSHAB	DBG\$CS_CR		

00000000G	00	57	DD	004CE	PUSHL	R7	:	
	09	03	FB	004D0	CALLS	#3, DBG\$NMATCH	:	
	6E	50	E9	004D7	BLBC	R0, 55\$	:	
		00	DD	004DA	MOVL	DBG\$SRC_RIGHT_MARGIN, RIGHT_MARGIN	:	0960
		18	11	004E1	BRB	58\$	:	
		OC	AC	DD 004E3	PUSHL	MESSAGE_VECT	:	0965
		04	AE	9F 004E6	PUSHAB	RIGHT_MARGIN	:	0964
		57	DD	004E9	PUSHL	R7	:	
00000000G	00	03	FB	004EB	CALLS	#3, DBG\$NSAVE_DECIMAL_INTEGER	:	
	06	50	E8	004F2	BLBS	R0, 58\$	:	
		0BE5	31	004F5	BRW	212\$	:	0967
	52	01	DD	004F8	MOVL	#1, LEFT_MARGIN	:	0976
	6E	52	D1	004FB	CMPL	LEFT_MARGIN, RIGHT_MARGIN	:	0984
		14	19	004FE	BLSS	59\$	:	
00000000G	00	00	FB	00500	CALLS	#0, DBG\$NSHOW_MARGINS	:	0987
		8F	DD	00507	PUSHL	#167440	:	0988
00000000G	00	01	FB	0050D	CALLS	#1, LIB\$SIGNAL	:	
	68	52	DD	00514	MOVL	LEFT_MARGIN, (NOUN_NODE)	:	0994
	OC	6E	DD	00517	MOVL	RIGHT_MARGIN, 12(NOUN_NODE)	:	0995
		30	11	0051B	BRB	61\$	:	0553
		03	DD	0051D	PUSHL	#3	:	0999
		EF	9F	0051F	PUSHAB	DBG\$CS_MAX_SOURCE_FILES	:	
		57	DD	00525	PUSHL	R7	:	
00000000G	00	03	FB	00527	CALLS	#3, DBG\$NMATCH	:	
	01	50	D1	0052E	CMPL	R0, #1	:	
		1D	12	00531	BNEQ	62\$	:	
	01	13	90	00533	MOVB	#19, 1(R9)	:	1003
		OC	AC	DD 00537	PUSHL	MESSAGE_VECT	:	1005
		08	AE	9F 0053A	PUSHAB	MAX_FILES	:	1004
		57	DD	0053D	PUSHL	R7	:	1005
00000000G	00	03	FB	0053F	CALLS	#3, DBG\$NSAVE_DECIMAL_INTEGER	:	
	AC	50	E9	00546	BLBC	R0, 56\$	:	
	68	04	AE	DD 00549	MOVL	MAX_FILES, (NOUN_NODE)	:	1008
		0B91	31	0054D	BRW	213\$	:	0553
		01	DD	00550	PUSHL	#1	:	1011
		EF	9F	00552	PUSHAB	DBG\$CS_MODE	:	
		57	DD	00558	PUSHL	R7	:	
00000000G	00	03	FB	0055A	CALLS	#3, DBG\$NMATCH	:	
	01	50	D1	00561	CMPL	R0, #1	:	
		03	13	00564	BEQL	63\$	:	
		0197	31	00566	BRW	87\$	:	
	01	06	90	00569	MOVB	#6, 1(R9)	:	1032
		01	DD	0056D	PUSHL	#1	:	1042
		EF	9F	0056F	PUSHAB	DBG\$CS_BINARY	:	
		57	DD	00575	PUSHL	R7	:	
00000000G	00	03	FB	00577	CALLS	#3, DBG\$NMATCH	:	
	01	50	D1	0057E	CMPL	R0, #1	:	
		06	12	00581	BNEQ	65\$	:	
	68	01	DD	00583	MOVL	#1, (NOUN_NODE)	:	1043
		0089	31	00586	BRW	70\$	:	
		03	DD	00589	PUSHL	#3	:	1045
		EF	9F	0058B	PUSHAB	DBG\$CS_DEFAULT	:	
		57	DD	00591	PUSHL	R7	:	
00000000G	00	03	FB	00593	CALLS	#3, DBG\$NMATCH	:	
	01	50	D1	0059A	CMPL	R0, #1	:	
		06	12	0059D	BNEQ	66\$	:	
	68	07	DD	0059F	MOVL	#7, (NOUN_NODE)	:	1046

		0089	31	005A2	BRW	72\$		
		01	DD	005A5	PUSHL	#1		1048
	00000000'	EF	9F	005A7	PUSHAB	DBG\$CS_DECIMAL		
		57	DD	005AD	PUSHL	R7		
00000000G	00	03	FB	005AF	CALLS	#3, DBG\$NMATCH		
	01	50	D1	005B6	CMPL	R0, #1		
		06	12	005B9	BNEQ	67\$		
	68	03	D0	005BB	MOVL	#3, (NOUN_NODE)		1049
		0088	31	005BE	BRW	74\$		
		01	DD	005C1	PUSHL	#1		1051
	00000000'	EF	9F	005C3	PUSHAB	DBG\$CS_HEXADECEMAL		
		57	DD	005C9	PUSHL	R7		
00000000G	00	03	FB	005CB	CALLS	#3, DBG\$NMATCH		
	01	50	D1	005D2	CMPL	R0, #1		
		06	12	005D5	BNEQ	68\$		
	68	04	D0	005D7	MOVL	#4, (NOUN_NODE)		1052
		0087	31	005DA	BRW	76\$		
		01	DD	005DD	PUSHL	#1		1054
	00000000'	EF	9F	005DF	PUSHAB	DBG\$CS_KEYPAD		
		57	DD	005E5	PUSHL	R7		
00000000G	00	03	FB	005E7	CALLS	#3, DBG\$NMATCH		
	01	50	D1	005EE	CMPL	R0, #1		
		06	12	005F1	BNEQ	69\$		
	68	0A	D0	005F3	MOVL	#10, (NOUN_NODE)		1055
		0086	31	005F6	BRW	78\$		
		03	DD	005F9	PUSHL	#3		1057
	00000000'	EF	9F	005FB	PUSHAB	DBG\$CS_NOKEYPAD		
		57	DD	00601	PUSHL	R7		
00000000G	00	03	FB	00603	CALLS	#3, DBG\$NMATCH		
	01	50	D1	0060A	CMPL	R0, #1		
		06	12	0060D	BNEQ	71\$		
	68	0B	D0	0060F	MOVL	#11, (NOUN_NODE)		1058
		0085	31	00612	BRW	80\$		
		05	DD	00615	PUSHL	#5		1060
	00000000'	EF	9F	00617	PUSHAB	DBG\$CS_NOSCREEN		
		57	DD	0061D	PUSHL	R7		
00000000G	00	03	FB	0061F	CALLS	#3, DBG\$NMATCH		
	01	50	D1	00626	CMPL	R0, #1		
		05	12	00629	BNEQ	73\$		
	68	09	D0	0062B	MOVL	#9, (NOUN_NODE)		1061
		19	11	0062E	BRB	74\$		
		01	DD	00630	PUSHL	#1		1063
	00000000'	EF	9F	00632	PUSHAB	DBG\$CS_NOSYMBOLS		
		57	DD	00638	PUSHL	R7		
00000000G	00	03	FB	0063A	CALLS	#3, DBG\$NMATCH		
	01	50	D1	00641	CMPL	R0, #1		
		05	12	00644	BNEQ	75\$		
	68	06	D0	00646	MOVL	#6, (NOUN_NODE)		1064
		6A	11	00649	BRB	82\$		
		01	DD	0064B	PUSHL	#1		1066
	00000000'	EF	9F	0064D	PUSHAB	DBG\$CS_OCTAL		
		57	DD	00653	PUSHL	R7		
00000000G	00	03	FB	00655	CALLS	#3, DBG\$NMATCH		
	01	50	D1	0065C	CMPL	R0, #1		
		05	12	0065F	BNEQ	77\$		
	68	02	D0	00661	MOVL	#2, (NOUN_NODE)		1067
		6D	11	00664	BRB	85\$		

		00000000'	03 DD 00666 77\$:	PUSHL #3		1069
			EF 9F 00668	PUSHAB DBG\$CS_SCREEN		
00000000G	00		57 DD 0066E	PUSHL R7		
	01		03 FB 00670	CALLS #3, DBG\$NMATCH		
			50 D1 00677	CMPL R0, #1		
	68		05 12 0067A	BNEQ 79\$		
			08 D0 0067C	MOVL #8, (NOUN_NODE)		1070
			52 11 0067F 78\$:	BRB 85\$		
			01 DD 00681 79\$:	PUSHL #1		1072
		00000000'	EF 9F 00683	PUSHAB DBG\$CS_SYMBOLS		
			57 DD 00689	PUSHL R7		
00000000G	00		03 FB 0068B	CALLS #3, DBG\$NMATCH		
	01		50 D1 00692	CMPL R0, #1		
			05 12 00695	BNEQ 81\$		
	68		05 D0 00697	MOVL #5, (NOUN_NODE)		1073
			37 11 0069A 80\$:	BRB 85\$		
			01 DD 0069C 81\$:	PUSHL #1		1075
		00000000'	EF 9F 0069E	PUSHAB DBG\$CS_G_FLOAT		
			57 DD 006A4	PUSHL R7		
00000000G	00		03 FB 006A6	CALLS #3, DBG\$NMATCH		
	01		50 D1 006AD	CMPL R0, #1		
			05 12 006B0	BNEQ 83\$		
	68		0C D0 006B2	MOVL #12, (NOUN_NODE)		1076
			1C 11 006B5 82\$:	BRB 85\$		
			03 DD 006B7 83\$:	PUSHL #3		1078
		00000000'	EF 9F 006B9	PUSHAB DBG\$CS_NOG_FLOAT		
			57 DD 006BF	PUSHL R7		
00000000G	00		03 FB 006C1	CALLS #3, DBG\$NMATCH		
	01		50 D1 006C8	CMPL R0, #1		
			03 13 006CB	BEQL 84\$		
	68		06DD 31 006CD	BRW 169\$		
			0D D0 006D0 84\$:	MOVL #13, (NOUN_NODE)		1079
			01 DD 006D3 85\$:	PUSHL #1		1099
		00000000'	EF 9F 006D5	PUSHAB DBG\$CS_COMMA		
			57 DD 006DB	PUSHL R7		
00000000G	00		03 FB 006DD	CALLS #3, DBG\$NMATCH		
	03		50 E8 006E4	BLBS R0, 86\$		
			053A 31 006E7	BRW 152\$		
	5B	08	A8 9E 006EA 86\$:	MOVAB 8(R8), LINK		1106
			04 DD 006EE	PUSHL #4		1107
00000000G	00		01 FB 006F0	CALLS #1, DBG\$GET_TEMPMEM		
	58		50 D0 006F7	MOVL R0, NOUN_NODE		
	6B		58 D0 006FA	MOVL NOUN_NODE, (LINK)		1108
			FE6D 31 006FD	BRW 64\$		1036
			04 DD 00700 87\$:	PUSHL #4		1119
		00000000'	EF 9F 00702	PUSHAB DBG\$CS_MODULE		
			57 DD 00708	PUSHL R7		
00000000G	00		03 FB 0070A	CALLS #3, DBG\$NMATCH		
	01		50 D1 00711	CMPL R0, #1		
			03 13 00714	BEQL 88\$		
			015A 31 00716	BRW 103\$		
			52 D4 00719 88\$:	CLRL ALLOCATE_FLAG		1131
			53 7C 0071B	CLRQ NOWAIT_FLAG		1132
			01 DD 0071D 89\$:	PUSHL #1		1137
		00000000'	EF 9F 0071F	PUSHAB DBG\$CS_SLASH		
			57 DD 00725	PUSHL R7		
00000000G	00		03 FB 00727	CALLS #3, DBG\$NMATCH		

55		50	E9	0072E	BLBC	R0, 94\$		
		04	DD	00731	PUSHL	#4		1144
	00000000'	EF	9F	00733	PUSHAB	DBG\$CS_ALLOCATE		
00000000G	00	57	DD	00739	PUSHL	R7		
	05	03	FB	0073B	CALLS	#3, DBG\$NMATCH		
	52	50	E9	00742	BLBC	R0, 90\$		1146
		01	DD	00745	MOVL	#1, ALLOCATE_FLAG		
		D3	11	00748	BRB	89\$		
	00000000'	01	DD	0074A	PUSHL	#1		1148
		EF	9F	0074C	PUSHAB	DBG\$CS_ALL		
00000000G	00	57	DD	00752	PUSHL	R7		
	05	03	FB	00754	CALLS	#3, DBG\$NMATCH		
	54	50	E9	0075B	BLBC	R0, 91\$		1150
		01	DD	0075E	MOVL	#1, ALL_FLAG		
	03	BA	11	00761	BRB	89\$		
	00000000'	EF	E8	00763	BLBS	DBG\$GL_DEVELOPER, 93\$		1152
		0640	31	0076A	BRW	169\$		
		01	DD	0076D	PUSHL	#1		1153
	00000000'	EF	9F	0076F	PUSHAB	DBG\$CS_NOWAIT		
00000000G	00	57	DD	00775	PUSHL	R7		
	E9	03	FB	00777	CALLS	#3, DBG\$NMATCH		
	53	50	E9	0077E	BLBC	R0, 92\$		1155
		01	DD	00781	MOVL	#1, NOWAIT_FLAG		
		97	11	00784	BRB	89\$		
04	A9	6342	3E	00786	MOVAV	(NOWAIT_FLAG)[ALLOCATE_FLAG], 4(R9)		1181
	07	54	E9	0078B	BLBC	ALL_FLAG, 95\$		1187
01	A9	08	90	0078E	MOVB	#8, 1(R9)		1189
		094C	31	00792	BRW	213\$		
	07	00000000G	00	91	00795	CMPB	DBG\$GB_LANGUAGE, #7	1215
			03	13	0079C	BEQL	96\$	
		008B	31	0079E	BRW	99\$		
	52	04	A7	DD	007A1	MOVL	4(R7), POINTER	1228
	56		67	3C	007A5	MOVZWL	(R7), LENGTH	1229
00000000G	00		52	D1	007A8	CMPL	POINTER, DBG\$GL_UPCASE_COMMAND_PTR	1230
			09	19	007AF	BLSS	97\$	
00000000G	00		52	D1	007B1	CMPL	POINTER, DBG\$GL_UPCASE_COMMAND_PTR+4	1231
			15	15	007B8	BLEQ	98\$	
	00000000'		EF	9F	007BA	PUSHAB	P.ACN	1233
			01	DD	007C0	PUSHL	#1	
	00028362		8F	DD	007C2	PUSHL	#164706	
00000000G	00		03	FB	007C8	CALLS	#3, LIB\$SIGNAL	
			A6	9E	007CF	MOVAB	3(R6), R0	1239
	50		04	C7	007D3	DIVL3	#4, R0, -(SP)	
7E	00000000G		01	FB	007D7	CALLS	#1, DBG\$GET_TEMP_MEM	
			50	DD	007DE	MOVL	R0, NEW_POINTER	
	5A		00	C2	007E1	SUBL2	DBG\$GL_UPCASE_COMMAND_PTR, R2	1240
50	52	00000000G	00	C1	007E8	ADDL3	DBG\$GL_ORIG_COMMAND_PTR, R2, TEMP_PTR	1241
6A	60	00000000G	56	28	007F0	MOVC3	LENGTH, (TEMP_PTR), (NEW_POINTER)	1242
	FF A64A		0D	90	007F4	MOVB	#13, -1(LENGTH)[NEW_POINTER]	1243
	OE AE	010E	8F	B0	007F9	MOVW	#270, STG_DESC+2	1248
	OC AE		56	B0	007FF	MOVW	LENGTH, STG_DESC	1249
	10 AE		5A	DD	00803	MOVL	NEW_POINTER, STG_DESC+4	1250
		14	AE	D4	00807	CLRL	STG_DESC+8	1251
		OC	AC	DD	0080A	PUSHL	MESSAGE_VECT	1257
			58	DD	0080D	PUSHL	NOUN_NODE	1256
		14	AE	9F	0080F	PUSHAB	STG_DESC	1255
00000000G	00		03	FB	00812	CALLS	#3, DBG\$NSAVE_STRING	1256

1D	50	E9	00819	BLBC	R0, 100\$	
50	OC	AE	3C 0081C	MOVZWL	STG DESC, R0	1264
50		56	C2 00820	SUBL2	LENGTH, R0	
67		50	A0 00823	ADDW2	R0, (R7)	
04	A7	50	C2 00826	SUBL2	R0, 4(R7)	1266
		13	11 0082A	BRB	101\$	1215
	OC	AC	DD 0082C	99\$:	PUSHL MESSAGE_VECT	1275
7E		57	7D 0082F	MOVQ	R7, -(SP)	
00000000G	00	03	FB 00832	CALLS	#3, DBG\$NSAVE_STRING	
03		50	E8 00839	100\$:	BLBS R0, 101\$	
	089E	31	0083C	BRW	212\$	
		01	DD 0083F	101\$:	PUSHL #1	1282
	00000000'	EF	9F 00841	PUSHAB	DBG\$CS_COMMA	
		57	DD 00847	PUSHL	R7	
00000000G	00	03	FB 00849	CALLS	#3, DBG\$NMATCH	
16		50	E9 00850	BLBC	R0, 102\$	
5B	08	A8	9E 00853	MOVAB	8(R8), LINK	1289
		04	DD 00857	PUSHL	#4	1290
00000000G	00	01	FB 00859	CALLS	#1, DBG\$GET_TEMPMEM	
58		50	D0 00860	MOVL	R0, NOUN_NODE	
6B		58	D0 00863	MOVL	NOUN_NODE, (LINK)	1291
	FF2C	31	00866	BRW	95\$	1207
	08	A8	D4 00869	102\$:	CLRL 8(NOUN_NODE)	1298
01	A9	07	90 0086C	MOVB	#7, 1(R9)	1299
	086E	31	00870	BRW	213\$	0553
		01	DD 00873	103\$:	PUSHL #1	1304
	00000000'	EF	9F 00875	PUSHAB	DBG\$CS_OUTPUT	
		57	DD 0087B	PUSHL	R7	
00000000G	00	03	FB 0087D	CALLS	#3, DBG\$NMATCH	
01		50	D1 00884	CMPL	R0, #1	
		03	13 00887	BEQL	104\$	
	010B	31	00889	BRW	119\$	
01	A9	09	90 0088C	104\$:	MOVB #9, 1(R9)	1333
		01	DD 00890	105\$:	PUSHL #1	1343
	00000000'	EF	9F 00892	PUSHAB	DBG\$CS_LOG	
		57	DD 00898	PUSHL	R7	
00000000G	00	03	FB 0089A	CALLS	#3, DBG\$NMATCH	
01		50	D1 008A1	CMPL	R0, #1	
		06	12 008A4	BNEQ	106\$	
68		01	D0 008A6	MOVL	#1, (NOUN_NODE)	1344
	0085	31	008A9	BRW	112\$	
		03	DD 008AC	106\$:	PUSHL #3	1346
	00000000'	EF	9F 008AE	PUSHAB	DBG\$CS_NOLOG	
		57	DD 008B4	PUSHL	R7	
00000000G	00	03	FB 008B6	CALLS	#3, DBG\$NMATCH	
01		50	D1 008BD	CMPL	R0, #1	
		05	12 008C0	BNEQ	107\$	
68		02	D0 008C2	MOVL	#2, (NOUN_NODE)	1347
		19	11 008C5	BRB	108\$	
	00000000'	01	DD 008C7	107\$:	PUSHL #1	1349
		EF	9F 008C9	PUSHAB	DBG\$CS_SCREEN_LOG	
		57	DD 008CF	PUSHL	R7	
00000000G	00	03	FB 008D1	CALLS	#3, DBG\$NMATCH	
01		50	D1 008D8	CMPL	R0, #1	
		05	12 008DB	BNEQ	109\$	
68		03	D0 008DD	MOVL	#3, (NOUN_NODE)	1350
		6A	11 008E0	108\$:	BRB 114\$	

		00000000'	03 DD 008E2 109\$:	PUSHL #3	1352
			EF 9F 008E4	PUSHAB DBG\$CS_NOSCREEN_LOG	
00000000G	00		57 DD 008EA	PUSHL R7	
	01		03 FB 008EC	CALLS #3, DBG\$NMATCH	
			50 D1 008F3	CMPL R0, #1	
	68		05 12 008F6	BNEQ 110\$	
			04 D0 008F8	MOVL #4, (NOUN_NODE)	1353
			6D 11 008FB	BRB 117\$	
		00000000'	01 DD 008FD 110\$:	PUSHL #1	1355
			EF 9F 008FF	PUSHAB DBG\$CS_TERMINAL	
00000000G	00		57 DD 00905	PUSHL R7	
	01		03 FB 00907	CALLS #3, DBG\$NMATCH	
			50 D1 0090E	CMPL R0, #1	
	68		05 12 00911	BNEQ 111\$	
			05 D0 00913	MOVL #5, (NOUN_NODE)	1356
			52 11 00916	BRB 117\$	
		00000000'	03 DD 00918 111\$:	PUSHL #3	1358
			EF 9F 0091A	PUSHAB DBG\$CS_NOTERMINAL	
00000000G	00		57 DD 00920	PUSHL R7	
	01		03 FB 00922	CALLS #3, DBG\$NMATCH	
			50 D1 00929	CMPL R0, #1	
	68		05 12 0092C	BNEQ 113\$	
			06 D0 0092E	MOVL #6, (NOUN_NODE)	1359
			37 11 00931 112\$:	BRB 117\$	
			01 DD 00933 113\$:	PUSHL #1	1361
		00000000'	EF 9F 00935	PUSHAB DBG\$CS_VERIFY	
00000000G	00		57 DD 0093B	PUSHL R7	
	01		03 FB 0093D	CALLS #3, DBG\$NMATCH	
			50 D1 00944	CMPL R0, #1	
	68		05 12 00947	BNEQ 115\$	
			07 D0 00949	MOVL #7, (NOUN_NODE)	1362
			1C 11 0094C 114\$:	BRB 117\$	
		00000000'	03 DD 0094E 115\$:	PUSHL #3	1364
			EF 9F 00950	PUSHAB DBG\$CS_NOVERIFY	
00000000G	00		57 DD 00956	PUSHL R7	
	01		03 FB 00958	CALLS #3, DBG\$NMATCH	
			50 D1 0095F	CMPL R0, #1	
			03 13 00962	BEQL 116\$	
	68	0446	31 00964	BRW 169\$	
			08 D0 00967 116\$:	MOVL #8, (NOUN_NODE)	1365
			01 DD 0096A 117\$:	PUSHL #1	1384
		00000000'	EF 9F 0096C	PUSHAB DBG\$CS_COMMA	
00000000G	00		57 DD 00972	PUSHL R7	
	03		03 FB 00974	CALLS #3, DBG\$NMATCH	
			50 E8 0097B	BLBS R0, 118\$	
	52	02A3	31 0097E	BRW 152\$	
		08	A8 9E 00981 118\$:	MOVAB 8(R8), LINK	1391
			04 DD 00985	PUSHL #4	1392
00000000G	00		01 FB 00987	CALLS #1, DBG\$GET_TEMPMEM	
	58		50 D0 0098E	MOVL R0, NOUN_NODE	
	62		58 D0 00991	MOVL NOUN_NODE, (LINK)	1393
		FEF9	31 00994	BRW 105\$	1338
			01 DD 00997 119\$:	PUSHL #1	1404
		00000000'	EF 9F 00999	PUSHAB DBG\$CS_RADIX	
00000000G	00		57 DD 0099F	PUSHL R7	
	01		03 FB 009A1	CALLS #3, DBG\$NMATCH	
			50 D1 009A8	CMPL R0, #1	

		03	13	009AB	BEQL	120\$		
		0149	31	009AD	BRW	140\$		
01	A9	1B	90	009B0	120\$: MOVB	#27, 1(R9)		1420
		01	DD	009B4	121\$: PUSHL	#1		1421
		00000000'	EF	9F	009B6	PUSHAB	DBG\$CS_SLASH	
		57	DD	009BC	PUSHL	R7		
00000000G	00	03	FB	009BE	CALLS	#3, DBG\$NMATCH		
	03	50	E8	009C5	BLBS	R0, 122\$		
		00A3	31	009C8	BRW	129\$		
		01	DD	009CB	122\$: PUSHL	#1		1425
		00000000'	EF	9F	009CD	PUSHAB	DBG\$CS_INPUT	
		57	DD	009D3	PUSHL	R7		
00000000G	00	03	FB	009D5	CALLS	#3, DBG\$NMATCH		
	01	50	D1	009DC	CMPL	R0, #1		
		06	12	009DF	BNEQ	124\$		
0C	AB	01	D0	009E1	MOVL	#1, 12(NOUN_NODE)		1426
		CD	11	009E5	123\$: BRB	121\$		
		02	DD	009E7	124\$: PUSHL	#2		1428
		00000000'	EF	9F	009E9	PUSHAB	DBG\$CS_OUTPUT	
		57	DD	009EF	PUSHL	R7		
00000000G	00	03	FB	009F1	CALLS	#3, DBG\$NMATCH		
	01	50	D1	009F8	CMPL	R0, #1		
		06	12	009FB	BNEQ	126\$		
0C	AB	02	D0	009FD	MOVL	#2, 12(NOUN_NODE)		1429
		B1	11	00A01	125\$: BRB	121\$		
		02	DD	00A03	126\$: PUSHL	#2		1431
		00000000'	EF	9F	00A05	PUSHAB	DBG\$CS_OVERRIDE	
		57	DD	00A0B	PUSHL	R7		
00000000G	00	03	FB	00A0D	CALLS	#3, DBG\$NMATCH		
	01	50	D1	00A14	CMPL	R0, #1		
		06	12	00A17	BNEQ	127\$		
01	A9	1C	90	00A19	MOVB	#28, 1(R9)		1432
		95	11	00A1D	BRB	121\$		
		01	DD	00A1F	127\$: PUSHL	#1		1439
		00000000'	EF	9F	00A21	PUSHAB	DBG\$CS_CR	
		57	DD	00A27	PUSHL	R7		
00000000G	00	03	FB	00A29	CALLS	#3, DBG\$NMATCH		
	0F	50	E9	00A30	BLBC	R0, 128\$		
		000280D0	8F	DD	00A33	PUSHL	#164048	1441
00000000G	00	01	FB	00A39	CALLS	#1, LIB\$SIGNAL		
		A3	11	00A40	BRB	123\$		
		57	DD	00A42	128\$: PUSHL	R7		1444
00000000G	00	01	FB	00A44	CALLS	#1, DBG\$NNEXT_WORD		
	0E	AE	8F	B0	00A4B	MOVW	#270, STG_DESC+2	1446
	0C	AE	60	9B	00A51	MOVZBW	(CS), STG_DESC	1447
	10	AE	A0	9E	00A55	MOVAB	1(R0), STG_DESC+4	1448
			AE	9F	00A5A	PUSHAB	STG_DESC	1449
		01	DD	00A5D	PUSHL	#1		
		00028238	8F	DD	00A5F	PUSHL	#164408	
00000000G	00	03	FB	00A65	CALLS	#3, LIB\$SIGNAL		
		93	11	00A6C	BRB	125\$		1422
		01	DD	00A6E	129\$: PUSHL	#1		1459
		00000000'	EF	9F	00A70	PUSHAB	DBG\$CS_BINARY	
		57	DD	00A76	PUSHL	R7		
00000000G	00	03	FB	00A78	CALLS	#3, DBG\$NMATCH		
	01	50	D1	00A7F	CMPL	R0, #1		
		05	12	00A82	BNEQ	131\$		

68	01	D0	00A84	130\$:	MOVL	#1, (NOUN_NODE)	1460	
	6D	11	00A87		BRB	139\$		
	03	DD	00A89	131\$:	PUSHL	#3	1462	
	00000000'	EF	9F	00A8B	PUSHAB	DBG\$CS_DEFAULT		
	57	DD	00A91		PUSHL	R7		
00000000G	00	03	FB	00A93	CALLS	#3, DBG\$NMATCH		
	01	50	D1	00A9A	CMPL	R0, #1		
		05	12	00A9D	BNEQ	133\$		
68	07	D0	00A9F	132\$:	MOVL	#7, (NOUN_NODE)	1463	
	52	11	00AA2		BRB	139\$		
	01	DD	00AA4	133\$:	PUSHL	#1	1465	
	00000000'	EF	9F	00AA6	PUSHAB	DBG\$CS_DECIMAL		
	57	DD	00AAC		PUSHL	R7		
00000000G	00	03	FB	00AAE	CALLS	#3, DBG\$NMATCH		
	01	50	D1	00AB5	CMPL	R0, #1		
		05	12	00AB8	BNEQ	135\$		
68	03	D0	00ABA	134\$:	MOVL	#3, (NOUN_NODE)	1466	
	37	11	00ABD		BRB	139\$		
	01	DD	00ABF	135\$:	PUSHL	#1	1468	
	00000000'	EF	9F	00AC1	PUSHAB	DBG\$CS_HEXADECIMAL		
	57	DD	00AC7		PUSHL	R7		
00000000G	00	03	FB	00AC9	CALLS	#3, DBG\$NMATCH		
	01	50	D1	00AD0	CMPL	R0, #1		
		05	12	00AD3	BNEQ	137\$		
68	04	D0	00AD5	136\$:	MOVL	#4, (NOUN_NODE)	1469	
	1C	11	00AD8		BRB	139\$		
	01	DD	00ADA	137\$:	PUSHL	#1	1471	
	00000000'	EF	9F	00ADC	PUSHAB	DBG\$CS_OCTAL		
	57	DD	00AE2		PUSHL	R7		
00000000G	00	03	FB	00AE4	CALLS	#3, DBG\$NMATCH		
	01	50	D1	00AEB	CMPL	R0, #1		
		03	13	00AEE	BEQL	138\$		
	02BA	31	00AF0		BRW	169\$		
68	02	D0	00AF3	138\$:	MOVL	#2, (NOUN_NODE)	1472	
	05E8	31	00AF6	139\$:	BRW	213\$		
	02	DD	00AF9	140\$:	PUSHL	#2	1490	
	00000000'	EF	9F	00AFB	PUSHAB	DBG\$CS_SCOPE		
	57	DD	00B01		PUSHL	R7		
00000000G	00	03	FB	00B03	CALLS	#3, DBG\$NMATCH		
	01	50	D1	00B0A	CMPL	R0, #1		
		61	12	00B0D	BNEQ	144\$		
01	A9	0A	90	00B0F	MOVB	#10, 1(R9)	1492	
		01	DD	00B13	PUSHL	#1	1497	
	00000000'	EF	9F	00B15	PUSHAB	DBG\$CS_SLASH		
	57	DD	00B1B		PUSHL	R7		
00000000G	00	03	FB	00B1D	CALLS	#3, DBG\$NMATCH		
	36	50	E9	00B24	BLBC	R0, 142\$		
		01	DD	00B27	PUSHL	#1	1499	
	00000000'	EF	9F	00B29	PUSHAB	DBG\$CS_MODULE		
	57	DD	00B2F		PUSHL	R7		
00000000G	00	03	FB	00B31	CALLS	#3, DBG\$NMATCH		
	06	50	E9	00B38	BLBC	R0, 141\$		
04	A9	01	D0	00B3B	MOVL	#1, 4(R9)	1501	
		1F	11	00B3F	BRB	143\$		
		57	DD	00B41	141\$:	PUSHL	R7	1503
00000000G	00	01	FB	00B43	CALLS	#1, DBG\$NNEXT_WORD		
		50	DD	00B4A	PUSHL	R0		

00000000G	00	00028A8A	01	DD	00B4C	PUSHL	#1	:	
			8F	DD	00B4E	PUSHL	#166538	:	
			03	FB	00B54	CALLS	#3, LIB\$SIGNAL	:	
			03	11	00B5B	BRB	143\$	:	1499
		04	A9	D4	00B5D	CLRL	4(R9)	:	1505
		0C	AC	DD	00B60	PUSHL	MESSAGE VECT	:	1508
			57	7D	00B63	MOVQ	R7, -(SP)	:	
00000000G	7E		03	FB	00B66	CALLS	#3, DBG\$NPARSE_SCOPE_LIST	:	
	00		F8DB	31	00B6D	BRW	50\$	:	
			02	DD	00B70	PUSHL	#2	:	1514
		00000000'	EF	9F	00B72	PUSHAB	DBG\$CS_SEARCH	:	
			57	DD	00B78	PUSHL	R7	:	
00000000G	00		03	FB	00B7A	CALLS	#3, DBG\$NMATCH	:	
	01		50	D1	00B81	CMPL	R0, #1	:	
			03	13	00B84	BEQL	145\$	:	
			00A1	31	00B86	BRW	153\$	:	
	01	A9	15	90	00B89	MOVB	#21, 1(R9)	:	1535
			01	DD	00B8D	PUSHL	#1	:	1544
		00000000'	EF	9F	00B8F	PUSHAB	DBG\$CS_ALL	:	
			57	DD	00B95	PUSHL	R7	:	
00000000G	00		03	FB	00B97	CALLS	#3, DBG\$NMATCH	:	
	01		50	D1	00B9E	CMPL	R0, #1	:	
			05	12	00BA1	BNEQ	147\$	:	
	68		01	D0	00BA3	MOVL	#1, (NOUN_NODE)	:	1545
			52	11	00BA6	BRB	151\$	:	
			01	DD	00BA8	PUSHL	#1	:	1547
		00000000'	EF	9F	00BAA	PUSHAB	DBG\$CS_IDENT	:	
			57	DD	00BB0	PUSHL	R7	:	
00000000G	00		03	FB	00BB2	CALLS	#3, DBG\$NMATCH	:	
	01		50	D1	00BB9	CMPL	R0, #1	:	
			05	12	00BBC	BNEQ	148\$	:	
	68		04	D0	00BBE	MOVL	#4, (NOUN_NODE)	:	1548
			37	11	00BC1	BRB	151\$	:	
			01	DD	00BC3	PUSHL	#1	:	1550
		00000000'	EF	9F	00BC5	PUSHAB	DBG\$CS_NEXT	:	
			57	DD	00BCB	PUSHL	R7	:	
00000000G	00		03	FB	00BCD	CALLS	#3, DBG\$NMATCH	:	
	01		50	D1	00BD4	CMPL	R0, #1	:	
			05	12	00BD7	BNEQ	149\$	:	
	68		02	D0	00BD9	MOVL	#2, (NOUN_NODE)	:	1551
			1C	11	00BDC	BRB	151\$	:	
			01	DD	00BDE	PUSHL	#1	:	1553
		00000000'	EF	9F	00BE0	PUSHAB	DBG\$CS_STRING	:	
			57	DD	00BE6	PUSHL	R7	:	
00000000G	00		03	FB	00BE8	CALLS	#3, DBG\$NMATCH	:	
	01		50	D1	00BEF	CMPL	R0, #1	:	
			03	13	00BF2	BEQL	150\$	:	
			01B6	31	00BF4	BRW	169\$	:	
	68		03	D0	00BF7	MOVL	#3, (NOUN_NODE)	:	1554
			01	DD	00BFA	PUSHL	#1	:	1574
		00000000'	EF	9F	00BFC	PUSHAB	DBG\$CS_COMMA	:	
			57	DD	00C02	PUSHL	R7	:	
00000000G	00		03	FB	00C04	CALLS	#3, DBG\$NMATCH	:	
	16		50	E9	00C0B	BLBC	R0, 152\$	:	
	52	08	A8	9E	00C0E	MOVAB	8(R8), LINK	:	1581
			04	DD	00C12	PUSHL	#4	:	1582
00000000G	00		01	FB	00C14	CALLS	#1, DBG\$GET_TEMPMEM	:	

58	50	D0	00C1B	MOVL	R0, NOUN_NODE		
62	58	D0	00C1E	MOVL	NOUN_NODE, (LINK)	1583	
	FF69	31	00C21	BRW	146\$	1539	
	08	A8	D4 00C24	CLRL	8(NOUN_NODE)	1590	
	04B7	31	00C27	BRW	213\$	0553	
		02	DD 00C2A	PUSHL	#2	1597	
	00000000'	EF	9F 00C2C	PUSHAB	DBG\$CS_SOURCE		
		57	DD 00C32	PUSHL	R7		
00000000G	00	03	FB 00C34	CALLS	#3, DBG\$NMATCH		
	01	50	D1 00C3B	CMPL	R0, #1		
		03	13 00C3E	BEQL	154\$		
		0086	31 00C40	BRW	159\$		
01	A9	12	90 00C43	MOVB	#18, 1(R9)	1599	
		01	DD 00C47	PUSHL	#1	1604	
	00000000'	EF	9F 00C49	PUSHAB	DBG\$CS_SLASH		
		57	DD 00C4F	PUSHL	R7		
00000000G	00	03	FB 00C51	CALLS	#3, DBG\$NMATCH		
	5E	50	E9 00C58	BLBC	R0, 158\$		
		04	DD 00C5B	PUSHL	#4	1617	
	00000000'	EF	9F 00C5D	PUSHAB	DBG\$CS_MODULE		
		57	DD 00C63	PUSHL	R7		
00000000G	00	03	FB 00C65	CALLS	#3, DBG\$NMATCH		
	11	50	E9 00C6C	BLBC	R0, 155\$		
		01	DD 00C6F	PUSHL	#1	1628	
	00000000'	EF	9F 00C71	PUSHAB	DBG\$CS_EQUAL		
		57	DD 00C77	PUSHL	R7		
00000000G	00	03	FB 00C79	CALLS	#3, DBG\$NMATCH		
	03	50	E8 00C80	BLBS	R0, 156\$		
		0441	31 00C83	BRW	210\$		
	0C	AC	DD 00C86	PUSHL	MESSAGE VECT	1639	
	0C	AE	9F 00C89	PUSHAB	MODNAMEPTR		
		57	DD 00C8C	PUSHL	R7		
00000000G	00	03	FB 00C8E	CALLS	#3, DBG\$NSAVE_STRING		
	03	50	E8 00C95	BLBS	R0, 157\$		
		0442	31 00C98	BRW	212\$		
	08	AE	DD 00C9B	PUSHL	MODNAMEPTR	1648	
00000000G	00	01	FB 00C9E	CALLS	#1, DBG\$STA_GETSOURCEMOD		
	04	50	D0 00CA5	MOVL	R0, 4(NOUN_NODE)		
		0E	12 00CA9	BNEQ	158\$	1654	
		08	AE	DD 00CAB	PUSHL	MODNAMEPTR	1658
		01	DD 00CAE	PUSHL	#1	1657	
	000281E8	8F	DD 00CB0	PUSHL	#164328		
		0226	31 00CB6	BRW	183\$		
	0C	AC	DD 00CB9	PUSHL	MESSAGE VECT	1668	
		57	7D 00CBC	MOVQ	R7, -(SP)		
00000000G	7E	03	FB 00CBF	CALLS	#3, DBG\$NGET_DIR_LIST		
	00	F782	31 00CC6	BRW	50\$		
		01	DD 00CC9	PUSHL	#1	1677	
	00000000'	EF	9F 00CCB	PUSHAB	DBG\$CS_STEP		
		57	DD 00CD1	PUSHL	R7		
00000000G	00	03	FB 00CD3	CALLS	#3, DBG\$NMATCH		
	01	50	D1 00CDA	CMPL	R0, #1		
		06	12 00CDD	BNEQ	160\$		
01	A9	0B	90 00CDF	MOVB	#11, 1(R9)	1679	
		74	11 00CE3	BRB	166\$	1680	
	04	00000000'	EF	E8 00CE5	BLBS	DBG\$GL_DEVELOPER, 161\$	1686
		50	D4 00CEC	CLRL	R0		

		11 11 00CEE	BRB	162\$	
		02 DD 00CF0	PUSHL	#2	1687
	00000000'	EF 9F 00CF2	PUSHAB	DBG\$CS_TASK	
00000000G	00	57 DD 00CF8	PUSHL	R7	
	01	03 FB 00CFA	CALLS	#3, DBG\$NMATCH	
		50 D1 00D01	CMPL	R0, #1	1686
		11 12 00D04	BNEQ	163\$	
01	A9	1E 90 00D06	MOVB	#30, 1(R9)	1689
	0280	8F BB 00D0A	PUSHR	#*M<R7,R9>	1690
00000000G	00	02 FB 00D0E	CALLS	#2, DBG\$NPARSE_SET_TASK	
		25 11 00D15	BRB	164\$	0553
		04 DD 00D17	PUSHL	#4	1696
	00000000'	EF 9F 00D19	PUSHAB	DBG\$CS_TERMINAL	
		57 DD 00D1F	PUSHL	R7	
00000000G	00	03 FB 00D21	CALLS	#3, DBG\$NMATCH	
	01	50 D1 00D28	CMPL	R0, #1	
		12 12 00D2B	BNEQ	165\$	
01	A9	19 90 00D2D	MOVB	#25, 1(R9)	1698
	0280	8F BB 00D31	PUSHR	#*M<R7,R9>	1699
00000000G	00	02 FB 00D35	CALLS	#2, DBG\$SCR_PARSE_SETTERM_CMD	
		03A2 31 00D3C	BRW	213\$	0553
		01 DD 00D3F	PUSHL	#1	1705
	00000000'	EF 9F 00D41	PUSHAB	DBG\$CS_TRACE	
		57 DD 00D47	PUSHL	R7	
00000000G	00	03 FB 00D49	CALLS	#3, DBG\$NMATCH	
	01	50 D1 00D50	CMPL	R0, #1	
		07 12 00D53	BNEQ	167\$	
01	A9	0E 90 00D55	MOVB	#14, 1(R9)	1707
		0332 31 00D59	BRW	205\$	1710
		02 DD 00D5C	PUSHL	#2	1715
	00000000'	EF 9F 00D5E	PUSHAB	DBG\$CS_TYPE	
		57 DD 00D64	PUSHL	R7	
00000000G	00	03 FB 00D66	CALLS	#3, DBG\$NMATCH	
	01	50 D1 00D6D	CMPL	R0, #1	
		03 13 00D70	BEQL	168\$	
		02FF 31 00D72	BRW	204\$	
		04 DD 00D75	PUSHL	#4	1746
00000000G	00	01 FB 00D77	CALLS	#1, DBG\$GET_TEMPMEM	
	52	50 D0 00D7E	MOVL	R0, LENGTH_NODE	
08	A8	52 D0 00D81	MOVL	LENGTH_NODE, 8(NOUN_NODE)	1747
		01 DD 00D85	PUSHL	#1	1751
	00000000'	EF 9F 00D87	PUSHAB	DBG\$CS_SLASH	
		57 DD 00D8D	PUSHL	R7	
00000000G	00	03 FB 00D8F	CALLS	#3, DBG\$NMATCH	
	41	50 E9 00D96	BLBC	R0, 173\$	
		01 DD 00D99	PUSHL	#1	1757
	00000000'	EF 9F 00D9B	PUSHAB	DBG\$CS_OVERRIDE	
		57 DD 00DA1	PUSHL	R7	
00000000G	00	03 FB 00DA3	CALLS	#3, DBG\$NMATCH	
	27	50 E8 00DAA	BLBS	R0, 172\$	
		01 DD 00DAD	PUSHL	#1	1761
	00000000'	EF 9F 00DAF	PUSHAB	DBG\$CS_CR	
		57 DD 00DB5	PUSHL	R7	
00000000G	00	03 FB 00DB7	CALLS	#3, DBG\$NMATCH	
	03	50 E8 00DBE	BLBS	R0, 170\$	
		0303 31 00DC1	BRW	210\$	
	000280D0	8F DD 00DC4	PUSHL	#164048	1763

00000000G	00	01	FB	00DCA	171\$:	CALLS	#1, DBG\$NMAKE_ARG_VECT	
		0305	31	00DD1		BRW	211\$	
	01	A9	0D	90	00DD4	172\$:	MOVB	#13, 1(R9)
			04	11	00DD8		BRB	174\$
	01	A9	0C	90	00DDA	173\$:	MOVB	#12, 1(R9)
			05	DD	00DDE	174\$:	PUSHL	#5
		00000000'	EF	9F	00DE0		PUSHAB	DBG\$CS_ASCIC
			57	DD	00DE6		PUSHL	R7
00000000G	00		03	FB	00DE8		CALLS	#3, DBG\$NMATCH
	53		50	D0	00DEF		MOVL	R0, R3
	01		53	D1	00DF2		CMPL	R3, #1
			16	13	00DF5		BEQL	175\$
		00000000'	02	DD	00DF7		PUSHL	#2
			EF	9F	00DF9		PUSHAB	DBG\$CS_AC
			57	DD	00DFF		PUSHL	R7
00000000G	00		03	FB	00E01		CALLS	#3, DBG\$NMATCH
	01		50	D1	00E08		CMPL	R0, #1
			06	12	00E0B		BNEQ	176\$
	68		26	D0	00E0D	175\$:	MOVL	#38, (NOUN_NODE)
		01E4	31	00E10		BRW	195\$	
			01	DD	00E13	176\$:	PUSHL	#1
		00000000'	EF	9F	00E15		PUSHAB	DBG\$CS_PACKED
			57	DD	00E1B		PUSHL	R7
00000000G	00		03	FB	00E1D		CALLS	#3, DBG\$NMATCH
	01		50	D1	00E24		CMPL	R0, #1
			51	12	00E27		BNEQ	180\$
	68		15	D0	00E29		MOVL	#21, (NOUN_NODE)
		00000000'	01	DD	00E2C		PUSHL	#1
			EF	9F	00E2E		PUSHAB	DBG\$CS_COLON
			57	DD	00E34		PUSHL	R7
00000000G	00		03	FB	00E36		CALLS	#3, DBG\$NMATCH
	0F		50	E8	00E3D		BLBS	R0, 177\$
		00028830	8F	DD	00E40		PUSHL	#165936
00000000G	00		01	FB	00E46		CALLS	#1, LIB\$SIGNAL
			28	11	00E4D		BRB	179\$
			52	DD	00E4F	177\$:	PUSHL	LENGTH_NODE
			57	DD	00E51		PUSHL	R7
00000000G	00		02	FB	00E53		CALLS	#2, DBG\$NSAVE_INTEGER
	58		50	E9	00E5A		BLBC	R0, 181\$
			62	D5	00E5D		TSTL	(LENGTH_NODE)
			05	19	00E5F		BLSS	178\$
	1F		62	D1	00E61		CMPL	(LENGTH_NODE), #31
			11	15	00E64		BLEQ	179\$
			62	DD	00E66	178\$:	PUSHL	(LENGTH_NODE)
			01	DD	00E68		PUSHL	#1
		00028828	8F	DD	00E6A		PUSHL	#165928
00000000G	00		03	FB	00E70		CALLS	#3, LIB\$SIGNAL
		0267	31	00E77	179\$:	BRW	213\$	
			01	DD	00E7A	180\$:	PUSHL	#1
		00000000'	EF	9F	00E7C		PUSHAB	DBG\$CS_ASCII
			57	DD	00E82		PUSHL	R7
00000000G	00		03	FB	00E84		CALLS	#3, DBG\$NMATCH
	01		50	D1	00E8B		CMPL	R0, #1
			59	12	00E8E		BNEQ	184\$
	68		0E	D0	00E90		MOVL	#14, (NOUN_NODE)
	62		04	D0	00E93		MOVL	#4, (LENGTH_NODE)
			01	DD	00E96		PUSHL	#1

		00000000'	EF	9F	00E98	PUSHAB	DBG\$CS_COLON		
			57	DD	00E9E	PUSHL	R7		
00000000G	00		03	FB	00EAO	CALLS	#3, DBG\$NMATCH		
	CD		50	E9	00EA7	BLBC	R0, 179\$		
			52	DD	00EAA	PUSHL	LENGTH_NODE	1843	
			57	DD	00EAC	PUSHL	R7		
00000000G	00		02	FB	00EAE	CALLS	#2, DBG\$NSAVE_INTEGER		
	03		50	E8	00EB5	BLBS	R0, 182\$		
		0222	31	00EB8	BRW	212\$			
		62	D5	00EBB	TSTL	(LENGTH_NODE)		1849	
		B8	12	00EBD	BNEQ	179\$			
00000000'	EF		01	B0	00EBF	MOVW	#1, ZERO_DESC	1857	
00000000'	EF	00000000'	EF	9E	00EC6	MOVAB	P.AEF, ZERO_DESC+4	1858	
		00000000'	EF	9F	00ED1	PUSHAB	ZERO_DESC	1859	
			01	DD	00ED7	PUSHL	#1		
		000281B0	8F	DD	00ED9	PUSHL	#164272		
00000000G	00		03	FB	00EDF	CALLS	#3, DBG\$NMAKE_ARG_VECT		
			01F0	31	00EE6	BRW	211\$		
			05	DD	00EE9	PUSHL	#5	1867	
		00000000'	EF	9F	00EEB	PUSHAB	DBG\$CS_ASCIZ		
			57	DD	00EF1	PUSHL	R7		
00000000G	00		03	FB	00EF3	CALLS	#3, DBG\$NMATCH		
	53		50	D0	00EFA	MOVL	R0, R3		
	01		53	D1	00EFD	CMPL	R3, #1		
			16	13	00F00	BEQL	185\$		
		00000000'	02	DD	00F02	PUSHL	#2	1868	
			EF	9F	00F04	PUSHAB	DBG\$CS_AZ		
			57	DD	00F0A	PUSHL	R7		
00000000G	00		03	FB	00F0C	CALLS	#3, DBG\$NMATCH		
	01		50	D1	00F13	CMPL	R0, #1		
			06	12	00F16	BNEQ	186\$		
	68		27	D0	00F18	MOVL	#39, (NOUN_NODE)	1870	
		00D9	31	00F1B	BRW	195\$		1871	
			01	DD	00F1E	PUSHL	#1	1874	
		00000000'	EF	9F	00F20	PUSHAB	DBG\$CS_BYTE		
			57	DD	00F26	PUSHL	R7		
00000000G	00		03	FB	00F28	CALLS	#3, DBG\$NMATCH		
	01		50	D1	00F2F	CMPL	R0, #1		
			09	12	00F32	BNEQ	187\$		
	68		06	D0	00F34	MOVL	#6, (NOUN_NODE)	1876	
	62		01	D0	00F37	MOVL	#1, (LENGTH_NODE)	1877	
		01A4	31	00F3A	BRW	213\$		1779	
			02	DD	00F3D	PUSHL	#2	1880	
		00000000'	EF	9F	00F3F	PUSHAB	DBG\$CS_DATE_TIME		
			57	DD	00F45	PUSHL	R7		
00000000G	00		03	FB	00F47	CALLS	#3, DBG\$NMATCH		
	01		50	D1	00F4E	CMPL	R0, #1		
			05	12	00F51	BNEQ	188\$		
	68		23	D0	00F53	MOVL	#35, (NOUN_NODE)	1882	
			68	11	00F56	BRB	192\$	1883	
			01	DD	00F58	PUSHL	#1	1886	
		00000000'	EF	9F	00F5A	PUSHAB	DBG\$CS_D_FLOAT		
			57	DD	00F60	PUSHL	R7		
00000000G	00		03	FB	00F62	CALLS	#3, DBG\$NMATCH		
	01		50	D1	00F69	CMPL	R0, #1		
			05	12	00F6C	BNEQ	189\$		
	68		0B	D0	00F6E	MOVL	#11, (NOUN_NODE)	1888	

		4D 11 00F71	BRB	192\$		1889
		01 DD 00F73	PUSHL	#1		1892
	00000000'	EF 9F 00F75	PUSHAB	DBG\$CS_FLOAT		
		57 DD 00F7B	PUSHL	R7		
00000000G	00	03 FB 00F7D	CALLS	#3, DBG\$NMATCH		
	53	50 D0 00F84	MOVL	R0, R3		
	01	53 D1 00F87	CMPL	R3, #1		
		16 13 00F8A	BEQL	190\$		
		01 DD 00F8C	PUSHL	#1		1893
	00000000'	EF 9F 00F8E	PUSHAB	DBG\$CS_F_FLOAT		
		57 DD 00F94	PUSHL	R7		
00000000G	00	03 FB 00F96	CALLS	#3, DBG\$NMATCH		
	01	50 D1 00F9D	CMPL	R0, #1		
		05 12 00FA0	BNEQ	191\$		
	68	0A D0 00FA2	MOVL	#10, (NOUN_NODE)		1895
		6D 11 00FA5	BRB	197\$		1896
		01 DD 00FA7	PUSHL	#1		1899
	00000000'	EF 9F 00FA9	PUSHAB	DBG\$CS_G_FLOAT		
		57 DD 00FAF	PUSHL	R7		
00000000G	00	03 FB 00FB1	CALLS	#3, DBG\$NMATCH		
	01	50 D1 00FB8	CMPL	R0, #1		
		06 12 00FBB	BNEQ	193\$		
	68	1B D0 00FBD	MOVL	#27, (NOUN_NODE)		1901
		008E 31 00FC0	BRW	201\$		1902
		01 DD 00FC3	PUSHL	#1		1905
	00000000'	EF 9F 00FC5	PUSHAB	DBG\$CS_H_FLOAT		
		57 DD 00FCB	PUSHL	R7		
00000000G	00	03 FB 00FCD	CALLS	#3, DBG\$NMATCH		
	01	50 D1 00FD4	CMPL	R0, #1		
		05 12 00FD7	BNEQ	194\$		
	68	1C D0 00FD9	MOVL	#28, (NOUN_NODE)		1907
		54 11 00FDC	BRB	199\$		1908
		01 DD 00FDE	PUSHL	#1		1911
	00000000'	EF 9F 00FE0	PUSHAB	DBG\$CS_INSTRUCTION		
		57 DD 00FE6	PUSHL	R7		
00000000G	00	03 FB 00FE8	CALLS	#3, DBG\$NMATCH		
	01	50 D1 00FEF	CMPL	R0, #1		
		07 12 00FF2	BNEQ	196\$		
	68	16 D0 00FF4	MOVL	#22, (NOUN_NODE)		1913
		62 D4 00FF7	CLRL	(LENGTH_NODE)		1914
		77 11 00FF9	BRB	203\$		1779
		01 DD 00FFB	PUSHL	#1		1917
	00000000'	EF 9F 00FFD	PUSHAB	DBG\$CS_LONGWORD		
		57 DD 01003	PUSHL	R7		
00000000G	00	03 FB 01005	CALLS	#3, DBG\$NMATCH		
	01	50 D1 0100C	CMPL	R0, #1		
		08 12 0100F	BNEQ	198\$		
	68	08 D0 01011	MOVL	#8, (NOUN_NODE)		1919
	62	04 D0 01014	MOVL	#4, (LENGTH_NODE)		1920
		59 11 01017	BRB	203\$		1779
		01 DD 01019	PUSHL	#1		1923
	00000000'	EF 9F 0101B	PUSHAB	DBG\$CS_OCTAWORD		
		57 DD 01021	PUSHL	R7		
00000000G	00	03 FB 01023	CALLS	#3, DBG\$NMATCH		
	01	50 D1 0102A	CMPL	R0, #1		
		09 12 0102D	BNEQ	200\$		
	68	1A D0 0102F	MOVL	#26, (NOUN_NODE)		1925

62	10	DO	01032	199\$:	MOVL	#16, (LENGTH_NODE)	1926
	008D	31	01035		BRW	209\$	1779
	01	DD	01038	200\$:	PUSHL	#1	1929
	00000000'	EF	9F 0103A		PUSHAB	DBG\$CS_QUADWORD	
	57	DD	01040		PUSHL	R7	
00000000G	00	03	FB 01042		CALLS	#3, DBG\$NMATCH	
	01	50	D1 01049		CMPL	R0, #1	
		08	12 0104C		BNEQ	202\$	
68	09	DO	0104E		MOVL	#9, (NOUN_NODE)	1931
62	08	DO	01051	201\$:	MOVL	#8, (LENGTH_NODE)	1932
	6F	11	01054		BRB	209\$	1779
	01	DD	01056	202\$:	PUSHL	#1	1935
	00000000'	EF	9F 01058		PUSHAB	DBG\$CS_WORD	
	57	DD	0105E		PUSHL	R7	
00000000G	00	03	FB 01060		CALLS	#3, DBG\$NMATCH	
	01	50	D1 01067		CMPL	R0, #1	
		45	12 0106A		BNEQ	207\$	
68	07	DO	0106C		MOVL	#7, (NOUN_NODE)	1937
62	02	DO	0106F		MOVL	#2, (LENGTH_NODE)	1938
	6D	11	01072	203\$:	BRB	213\$	1779
	01	DD	01074	204\$:	PUSHL	#1	1959
	00000000'	EF	9F 01076		PUSHAB	DBG\$CS_WATCH	
	57	DD	0107C		PUSHL	R7	
00000000G	00	03	FB 0107E		CALLS	#3, DBG\$NMATCH	
	01	50	D1 01085		CMPL	R0, #1	
		13	12 01088		BNEQ	206\$	
01	A9	11	90 0108A		MOVB	#17, 1(R9)	1961
		AC	DD 0108E	205\$:	PUSHL	MESSAGE_VECT	1962
	0C	8F	BB 01091		PUSHR	#^M<R7,R9>	
00000000G	00	03	FB 01095		CALLS	#3, DBG\$EVENT_SYNTAX	
		04	0109C		RET		
		03	DD 0109D	206\$:	PUSHL	#3	1968
	00000000'	EF	9F 0109F		PUSHAB	DBG\$CS_WINDOW	
	57	DD	010A5		PUSHL	R7	
00000000G	00	03	FB 010A7		CALLS	#3, DBG\$NMATCH	
	01	50	D1 010AE		CMPL	R0, #1	
		03	13 010B1	207\$:	BEQL	208\$	
		FCF7	31 010B3		BRW	169\$	
01	A9	1A	90 010B6	208\$:	MOVB	#26, 1(R9)	1970
	0280	8F	BB 010BA		PUSHR	#^M<R7,R9>	1971
00000000G	00	02	FB 010BE		CALLS	#2, DBG\$SCR_PARSE_SETWIND_CMD	
		1A	11 010C5	209\$:	BRB	213\$	0553
		57	DD 010C7	210\$:	PUSHL	R7	1984
00000000G	00	01	FB 010C9		CALLS	#1, DBG\$NNEXT_WORD	
		50	DD 010D0		PUSHL	R0	
00000000G	00	01	FB 010D2		CALLS	#1, DBG\$NSYNTAX_ERROR	
	0C	50	DO 010D9	211\$:	MOVL	R0, @MESSAGE_VECT	
	50	04	DO 010DD	212\$:	MOVL	#4, R0	1986
		04	010E0		RET		
	50	01	DO 010E1	213\$:	MOVL	#1, R0	1994
		04	010E4		RET		1996

; Routine Size: 4325 bytes, Routine Base: DBG\$CODE + 0000


```
: 1867      1997 1 GLOBAL ROUTINE DBG$NEXECUTE_SET (VERB_NODE, MESSAGE_VECT) =
: 1868      1998 1
: 1869      1999 1 FUNCTIONAL DESCRIPTION:
: 1870      2000 1
: 1871      2001 1     This routine processes the command execution tree corresponding to the parsed
: 1872      2002 1     SET ... command. Appropriate semantic actions are taken.
: 1873      2003 1
: 1874      2004 1 FORMAL PARAMETERS:
: 1875      2005 1
: 1876      2006 1     VERB_NODE -           The head node in the command execution tree
: 1877      2007 1
: 1878      2008 1     MESSAGE_VECT -        The address of a longword to contain the address
: 1879      2009 1     of a message argument vector
: 1880      2010 1
: 1881      2011 1 IMPLICIT INPUTS:
: 1882      2012 1
: 1883      2013 1     DBG$GL_LOG_BUF -        Version 2 debugger pointer to counted string filespec
: 1884      2014 1
: 1885      2015 1     DBG$GL_CONTEXT -        " " " context switch vector for log and output
: 1886      2016 1
: 1887      2017 1     DBG$GB_DEF_OUT -        " " " output control vector
: 1888      2018 1
: 1889      2019 1     DBG$GL_LOGFAB -        " " " FAB for log file
: 1890      2020 1
: 1891      2021 1     DBG$GL_LOGRAB -        " " " RAB " " "
: 1892      2022 1
: 1893      2023 1 IMPLICIT OUTPUTS:
: 1894      2024 1
: 1895      2025 1     Any of the above implicit inputs may be altered
: 1896      2026 1
: 1897      2027 1 ROUTINE VALUE:
: 1898      2028 1
: 1899      2029 1     An unsigned integer longword completion code
: 1900      2030 1
: 1901      2031 1 COMPLETION CODES:
: 1902      2032 1
: 1903      2033 1     STSSK_SEVERE (4)          - Signifies that the command was not executed
: 1904      2034 1
: 1905      2035 1     STSSK_SUCCESS (1)         - Signifies that a command was executed.
: 1906      2036 1
: 1907      2037 1 SIDE EFFECTS:
: 1908      2038 1
: 1909      2039 1     Various semantic actions are performed on data structures. That is, the
: 1910      2040 1     state of the debugger is altered.
: 1911      2041 1
: 1912      2042 1
: 1913      2043 2 BEGIN
: 1914      2044 2
: 1915      2045 2 MAP
: 1916      2046 2     VERB_NODE : REF DBG$VERB_NODE; ! Pointer to command Verb Node
: 1917      2047 2
: 1918      2048 2 LOCAL
: 1919      2049 2     NOUN_NODE : REF DBG$NOUN_NODE; ! Pointer to current Noun Node
: 1920      2050 2
: 1921      2051 2
: 1922      2052 2
: 1923      2053 2 ! Recover the noun node.
```



```

: 1924      2054      2      !
: 1925      2055      2      NOUN_NODE = .VERB_NODE [DBG$L_VERB_OBJECT_PTR];
: 1926      2056      2
: 1927      2057      2
: 1928      2058      2      ! Transfer to a subnetwork on the basis of the composite verb.
: 1929      2059      2
: 1930      2060      2      CASE .VERB_NODE[DBG$B_VERB_COMPOSITE] FROM SET_MINIMUM TO SET_MAXIMUM OF
: 1931      2061      2      SET
: 1932      2062      2
: 1933      2063      2
: 1934      2064      2      ! Execute the SET BREAK command.
: 1935      2065      2
: 1936      2066      2      [SET BREAK]:
: 1937      2067      2      DBG$EVENT_SEMANTICS (.VERB_NODE, .MESSAGE_VECT);
: 1938      2068      2
: 1939      2069      2
: 1940      2070      2      ! Execute the SET DEVELOPER command by setting the specified
: 1941      2071      2      bits in the DBG$GL_DEVELOPER longword.
: 1942      2072      2
: 1943      2073      2      [SET DEVELOPER] :
: 1944      2074      3      BEGIN
: 1945      2075      3      IF .VERB_NODE[DBG$L_VERB_OBJECT_PTR] NEQ 0
: 1946      2076      3      THEN
: 1947      2077      4      BEGIN
: 1948      2078      4      NOUN_NODE = .VERB_NODE[DBG$L_VERB_OBJECT_PTR];
: 1949      2079      4      WHILE TRUE DO
: 1950      2080      5      BEGIN
: 1951      2081      5      DBG$GL_DEVELOPER[.NOUN_NODE [DBG$L_NOUN_VALUE]] = TRUE;
: 1952      2082      5      IF .NOUN_NODE[DBG$L_NOUN_LINK] EQL 0 THEN EXITLOOP;
: 1953      2083      5      NOUN_NODE = .NOUN_NODE[DBG$L_NOUN_LINK];
: 1954      2084      4      END;
: 1955      2085      4
: 1956      2086      3      END;
: 1957      2087      3
: 1958      2088      2      END;
: 1959      2089      2
: 1960      2090      2
: 1961      2091      2      ! Execute the SET DEFINE command.
: 1962      2092      2
: 1963      2093      2      [SET DEFINE] :
: 1964      2094      3      BEGIN
: 1965      2095      3      DBG$SET_DEFINE_LVL (USER_DEF_DEFINE);
: 1966      2096      3      DBG$GB_DEFINE_PTR[DEFINE_ONLY] = .NOUN_NODE [DBG$L_NOUN_VALUE];
: 1967      2097      3      END;
: 1968      2098      2
: 1969      2099      2
: 1970      2100      2      ! Execute the SET DISPLAY command.
: 1971      2101      2
: 1972      2102      2      [SET DISPLAY]:
: 1973      2103      2      DBG$SCR_EXECUTE_DISPLAY_CMD(.VERB_NODE, TRUE);
: 1974      2104      2
: 1975      2105      2
: 1976      2106      2      ! Execute the SET EXCEPTION BREAK command.
: 1977      2107      2
: 1978      2108      2      [set_exception_break] :
: 1979      2109      3      BEGIN
: 1980      2110      3      dbg$gb_resignal = false;
```



```

: 1981      2111      3
: 1982      2112
: 1983      2113
: 1984      2114
: 1985      2115
: 1986      2116
: 1987      2117
: 1988      2118
: 1989      2119
: 1990      2120
: 1991      2121
: 1992      2122
: 1993      2123
: 1994      2124
: 1995      2125
: 1996      2126
: 1997      2127
: 1998      2128
: 1999      2129
: 2000      2130
: 2001      2131
: 2002      2132
: 2003      2133
: 2004      2134
: 2005      2135
: 2006      2136
: 2007      2137
: 2008      2138
: 2009      2139
: 2010      2140
: 2011      2141
: 2012      2142
: 2013      2143
: 2014      2144
: 2015      2145
: 2016      2146
: 2017      2147
: 2018      2148
: 2019      2149
: 2020      2150
: 2021      2151
: 2022      2152
: 2023      2153
: 2024      2154
: 2025      2155
: 2026      2156
: 2027      2157
: 2028      2158
: 2029      2159
: 2030      2160
: 2031      2161
: 2032      2162
: 2033      2163
: 2034      2164
: 2035      2165
: 2036      2166
: 2037      2167

RETURN DBG$EVENT_SEMANTICS (.VERB_NODE, .MESSAGE_VECT);
END;

! Execute the SET KEY command.
[SET KEY] :
BEGIN
    LOCAL
        OUTPUT_LOG,
        SET_STATUS,
        DESC_PTR: REF DBG$STG_DESC;
    OUTPUT_LOG = .VERB_NODE [DBG$L_VERB_ADVERB_PTR];

    ! Make the call to set the status.
    SET_STATUS = SMG$SET_DEFAULT_STATE (DBG$GL_KEY_TABLE_ID,
                                         .NOUN_NODE [DBG$L_NOUN_VALUE]);
    IF NOT SET_STATUS THEN SIGNAL(DBG$_SETKEYERR);
    IF .OUTPUT_LOG
    THEN
        SIGNAL(DBG$_SETKEY, 1, .NOUN_NODE [DBG$L_NOUN_VALUE]);
    END;

[SET LANGUAGE] :
BEGIN
    DBG$SET_LANG (0, .NOUN_NODE [DBG$L_NOUN_VALUE]);
END;

[set log] :
BEGIN
    dbg$gl_log_buf = .noun_node [dbg$l_noun_value];
    IF NOT dbg$inset_log (.message_vect)
    THEN
        RETURN sts$k_severe;
    END;

[set margins] :
BEGIN
    dbg$src_left_margin = .noun_node[dbg$l_noun_value];
    dbg$src_right_margin = .noun_node[dbg$l_noun_value2];
END;

[set max source_files] :
BEGIN
    dbg$src_set_max_files(.noun_node[dbg$l_noun_value]);
END;

! Execute the SET MODE command.
[SET MODE] :
BEGIN
```


2038	2168	3
2039	2169	3
2040	2170	3
2041	2171	3
2042	2172	3
2043	2173	3
2044	2174	3
2045	2175	3
2046	2176	3
2047	2177	3
2048	2178	4
2049	2179	4
2050	2180	4
2051	2181	4
2052	2182	4
2053	2183	4
2054	2184	5
2055	2185	5
2056	2186	5
2057	2187	4
2058	2188	4
2059	2189	4
2060	2190	5
2061	2191	5
2062	2192	5
2063	2193	4
2064	2194	4
2065	2195	4
2066	2196	5
2067	2197	5
2068	2198	5
2069	2199	4
2070	2200	4
2071	2201	4
2072	2202	5
2073	2203	5
2074	2204	5
2075	2205	5
2076	2206	5
2077	2207	4
2078	2208	4
2079	2209	4
2080	2210	5
2081	2211	5
2082	2212	5
2083	2213	4
2084	2214	4
2085	2215	4
2086	2216	4
2087	2217	4
2088	2218	4
2089	2219	4
2090	2220	4
2091	2221	4
2092	2222	4
2093	2223	4
2094	2224	4

```
! Set the mode level to the correct level.
```

```
DBG$SET_MOD_LVL (USER_DEF_MODE);
```

```
! Now fill in the correct mode values.
```

```
WHILE .NOUN_NODE NEQA 0 DO
```

```
  BEGIN
```

```
    CASE .NOUN_NODE [DBG$NOUN_VALUE]
```

```
      FROM MODE_LOWEST TO MODE_HIGHEST OF
```

```
        SET
```

```
        [MODE_BINARY] :
```

```
          BEGIN
```

```
            DBG$GB_RADIX[DBG$B_RADIX_INPUT] = DBG$K_BINARY;
```

```
            DBG$GB_RADIX[DBG$B_RADIX_OUTPUT] = DBG$R_BINARY;
```

```
          END;
```

```
        [MODE_OCTAL] :
```

```
          BEGIN
```

```
            DBG$GB_RADIX[DBG$B_RADIX_INPUT] = DBG$K_OCTAL;
```

```
            DBG$GB_RADIX[DBG$B_RADIX_OUTPUT] = DBG$R_OCTAL;
```

```
          END;
```

```
        [MODE_DECIMAL] :
```

```
          BEGIN
```

```
            DBG$GB_RADIX[DBG$B_RADIX_INPUT] = DBG$K_DECIMAL;
```

```
            DBG$GB_RADIX[DBG$B_RADIX_OUTPUT] = DBG$R_DECIMAL;
```

```
          END;
```

```
        [MODE_DEFAULT] :
```

```
          BEGIN
```

```
            DBG$GB_RADIX[DBG$B_RADIX_INPUT] =
```

```
              DBG$NGET_TRANS_RADIX(DBG$K_DEFAULT);
```

```
            DBG$GB_RADIX[DBG$B_RADIX_OUTPUT] =
```

```
              DBG$NGET_TRANS_RADIX(DBG$K_DEFAULT);
```

```
          END;
```

```
        [MODE_HEX] :
```

```
          BEGIN
```

```
            DBG$GB_RADIX[DBG$B_RADIX_INPUT] = DBG$K_HEX;
```

```
            DBG$GB_RADIX[DBG$B_RADIX_OUTPUT] = DBG$R_HEX;
```

```
          END;
```

```
        [MODE_SYMBOLS] :
```

```
          DBG$GB_MOD_PTR [MODE_SYMBOLS] = TRUE;
```

```
        [MODE_NOSYMBOLS] :
```

```
          DBG$GB_MOD_PTR [MODE_SYMBOLS] = FALSE;
```

```
        [MODE_G_FLOAT] :
```

```
          DBG$GB_MOD_PTR [MODE_G_FLOATS] = TRUE;
```

```
        [MODE_NOG_FLOAT] :
```



```
2095 2225 4          DBG$GB_MOD_PTR [MODE_G_FLOATS] = FALSE;
2096 2226 4
2097 2227 4      [MODE_SCREEN]:
2098 2228 4          DBG$SCR_SCREEN_MODE(TRUE);
2099 2229 4
2100 2230 4      [MODE_NOSCREEN]:
2101 2231 4          DBG$SCR_SCREEN_MODE(FALSE);
2102 2232 4
2103 2233 4      [MODE_KEYPAD]:
2104 2234 5          BEGIN
2105 2235 5          DBG$GB_KEYPAD_INPUT = TRUE;
2106 2236 5          SMG$SET_KEYPAD_MODE (DBG$GL_KEYBOARD_ID,%REF(1));
2107 2237 4          END;
2108 2238 4
2109 2239 4      [MODE_NOKEYPAD]:
2110 2240 5          BEGIN
2111 2241 5          DBG$GB_KEYPAD_INPUT = FALSE;
2112 2242 5          SMG$SET_KEYPAD_MODE (DBG$GL_KEYBOARD_ID,%REF(0));
2113 2243 4          END;
2114 2244 4
2115 2245 4      TES;
2116 2246 4
2117 2247 4
2118 2248 4      ! Recover the next Noun Node and loop.
2119 2249 4      !
2120 2250 4      NOUN_NODE = .NOUN_NODE [DBG$L_NOUN_LINK];
2121 2251 4
2122 2252 3      END;          ! End of loop over SET MODE keywords
2123 2253 3
2124 2254 2      END;          ! End of SET MODE command
2125 2255 2
2126 2256 2
2127 2257 2      ! Handle the SET MODULE command.
2128 2258 2      !
2129 2259 2      [SET_MODULE, SET_MODULE_ALL] :
2130 2260 2      BEGIN
2131 2261 2
2132 2262 2      EXTERNAL ROUTINE
2133 2263 2          LIB$SPAWN,
2134 2264 2          SYSS$WAITFR;
2135 2265 2
2136 2266 2      LOCAL
2137 2267 2          ALLOCATE_FLAG,
2138 2268 2          EVNT_FLAG,
2139 2269 2          FLAGS,
2140 2270 2          NAME_BUFF: REF VECTOR [,BYTE],      ! Module name buffer
2141 2271 2          STATUS;
2142 2272 2
2143 2273 2
2144 2274 2      ! Before doing anything, check for /NOWAIT flag.
2145 2275 2      ! The ADVERB_POINTER field indicates whether /NOWAIT
2146 2276 2      ! was specified and/or whether /ALLOCATE was specified.
2147 2277 2      ! The codes are
2148 2278 2      ! 0 - neither /NOWAIT or /ALLOCATE
2149 2279 2      ! 1 - /NOWAIT, not /ALLOCATE
2150 2280 2      ! 2 - not /NOWAIT, /ALLOCATE
2151 2281 2      ! 3 - both were specified
```



```

: 2152      2282      3
: 2153      2283      3
: 2154      2284      3
: 2155      2285      3
: 2156      2286      4
: 2157      2287      4
: 2158      2288      4
: 2159      2289      4
: 2160      2290      4
: 2161      2291      4
: 2162      2292      4
: 2163      2293      4
: 2164      2294      4
: 2165      2295      4
: 2166      2296      4
: 2167      2297      4
: 2168      2298      4
: 2169      2299      4
: 2170      2300      4
: 2171      2301      3
: 2172      2302      3
: 2173      2303      3
: 2174      2304      3
: 2175      2305      3
: 2176      2306      3
: 2177      2307      3
: 2178      2308      3
: 2179      2309      3
: 2180      2310      3
: 2181      2311      3
: 2182      2312      3
: 2183      2313      3
: 2184      2314      3
: 2185      2315      3
: 2186      2316      3
: 2187      2317      4
: 2188      2318      4
: 2189      2319      4
: 2190      2320      4
: 2191      2321      4
: 2192      2322      4
: 2193      2323      4
: 2194      2324      4
: 2195      2325      4
: 2196      2326      5
: 2197      2327      5
: 2198      2328      5
: 2199      2329      5
: 2200      2330      5
: 2201      2331      4
: 2202      2332      4
: 2203      2333      4
: 2204      2334      4
: 2205      2335      4
: 2206      2336      3
: 2207      2337      3
: 2208      2338      3

!
! STATUS = FALSE;
! IF .VERB_NODE [DBG$L_VERB_ADVERB_PTR]
! THEN
!     BEGIN
!
!         ! We accomplish SET MODU/NOWAIT by spawning a subprocess
!         ! with the /NOWAIT flag passed to LIB$SPAWN. The
!         ! subprocess can accept DCL commands until the
!         ! SET MODU completes. We pass in an event flag number (0)
!         ! to be set when the subprocess completes.
!
!         FLAGS = 1;
!         EVNT_FLAG = 0;
!         STATUS = LIB$SPAWN(0,0,0,FLAGS,0,0,0,EVNT_FLAG);
!         IF NOT .STATUS
!         THEN
!             SIGNAL(DBG$_NOSPAWNI,0,.STATUS);
!         END;
!
!     ! SET MODULE/ALLOCATE was specified if the 2nd bit of
!     ! the ADVERB_PTR field is set.
!     ALLOCATE_FLAG = .VERB_NODE[DBG$L_VERB_ADVERB_PTR]/2;
!
!     ! Loop through the list of modules to set.
!     IF .verb_node[dbg$b_verb_composite] EQL set_module_all
!     THEN
!         dbg$rst_setmod (0, 0, .allocate_flag)
!     ELSE
!         WHILE .noun_node NEQA 0
!         DO
!             BEGIN
!                 ! Retrieve the name buffer and call the symbol table
!                 ! Module names are stored away as counted strings
!
!                 name_buff = .noun_node [dbg$l_noun_value];
!                 IF NOT dbg$rst_setmod (name_buff [1], .name_buff [0],
!                                     .allocate_flag)
!                 THEN
!                     BEGIN
!                         .message_vect = dbg$nmake_arg_vect (dbg$_nosuchmodu,
!                                                             1,
!                                                             name_buff [0]);
!
!                         RETURN sts$k_severe;
!                     END;
!
!                 ! Obtain the next noun node
!                 !
!                 noun_node = .noun_node [dbg$l_noun_link];
!             END;
!             ! End of Loop
!
!         ! Wait for spawned subprocess to complete if there was one.
```



```
.. 2209      2339      3
.. 2210      2340      3
.. 2211      2341      3
.. 2212      2342      4
.. 2213      2343      4
.. 2214      2344      4
.. 2215      2345      3
.. 2216      2346      2
.. 2217      2347      2
.. 2218      2348      2
.. 2219      2349      2
.. 2220      2350      2
.. 2221      2351      2
.. 2222      2352      2
.. 2223      2353      2
.. 2224      2354      2
.. 2225      2355      2
.. 2226      2356      2
.. 2227      2357      2
.. 2228      2358      2
.. 2229      2359      2
.. 2230      2360      2
.. 2231      2361      2
.. 2232      2362      3
.. 2233      2363      4
.. 2234      2364      4
.. 2235      2365      4
.. 2236      2366      4
.. 2237      2367      4
.. 2238      2368      4
.. 2239      2369      5
.. 2240      2370      5
.. 2241      2371      5
.. 2242      2372      5
.. 2243      2373      5
.. 2244      2374      5
.. 2245      2375      5
.. 2246      2376      5
.. 2247      2377      5
.. 2248      2378      5
.. 2249      2379      5
.. 2250      2380      5
.. 2251      2381      5
.. 2252      2382      5
.. 2253      2383      5
.. 2254      2384      4
.. 2255      2385      4
.. 2256      2386      4
.. 2257      2387      5
.. 2258      2388      5
.. 2259      2389      5
.. 2260      2390      5
.. 2261      2391      5
.. 2262      2392      5
.. 2263      2393      5
.. 2264      2394      4
.. 2265      2395      4
```

```
!
if .status
THEN
BEGIN
  SIGNAL(DBG$_SETMODU);
  SYSSWAITFR(.EVNT_FLAG);
END;
END;

[set_output] :
BEGIN
  LITERAL
    NOUN_LITERAL_LOG           = 1,
    NOUN_LITERAL_NOLOG         = 2,
    NOUN_LITERAL_SCREEN        = 3,
    NOUN_LITERAL_NOSCREEN      = 4,
    NOUN_LITERAL_TERMINAL       = 5,
    NOUN_LITERAL_NOTERMINAL     = 6,
    NOUN_LITERAL_VERIFY         = 7,
    NOUN_LITERAL_NOVERIFY       = 8;

  ! Add the user defined output setting

  WHILE TRUE DO
  BEGIN
    CASE .NOUN_NODE(DBG$_NOUN_VALUE) FROM NOUN_LITERAL_LOG
    TO NOUN_LITERAL_NOVERIFY OF

    SET

    [NOUN_LITERAL_LOG] :
    BEGIN
      LOCAL
        STATUS,
        B_TYPE;

      STATUS = DBG$NSETUP_LOG (B_TYPE);
      IF NOT .STATUS
      THEN
        OUTPUT_RMS_ERROR (.B_TYPE);

      DBG$GB_DEF_OUT [OUT_LOG] = TRUE;
      IF .DBG$GB_DEF_OUT[OUT_SCREEN]
      THEN
        DBG$GL_SCREEN_LOG = TRUE;

      END;

    [NOUN_LITERAL_NOLOG] :
    BEGIN
      IF NOT .DBG$GB_DEF_OUT [OUT_TERM]
      THEN
        DBG$NOUT_INFO (DBG$_OUTPUTLOST);

      DBG$GB_DEF_OUT [OUT_LOG] = FALSE;
      DBG$GL_SCREEN_LOG = FALSE;

      END;
```



```
2266 2396 4 [NOUN_LITERAL_SCREEN]:  
2267 2397 5 BEGIN  
2268 2398 5 DBG$GB_DEF_OUT[OUT_SCREEN] = TRUE;  
2269 2399 5 IF .DBG$GB_DEF_OUT[OUT_LOG]  
2270 2400 5 THEN  
2271 2401 5     DBG$GL_SCREEN_LOG = TRUE;  
2272 2402 5  
2273 2403 4 END;  
2274 2404 4  
2275 2405 4 [NOUN_LITERAL_NOSCREEN]:  
2276 2406 5 BEGIN  
2277 2407 5 DBG$GB_DEF_OUT[OUT_SCREEN] = FALSE;  
2278 2408 5 DBG$GL_SCREEN_LOG = FALSE;  
2279 2409 4 END;  
2280 2410 4  
2281 2411 4 [NOUN_LITERAL_TERMINAL] :  
2282 2412 5 BEGIN  
2283 2413 5 DBG$GB_DEF_OUT [OUT_TERM] = TRUE;  
2284 2414 4 END;  
2285 2415 4  
2286 2416 4 [NOUN_LITERAL_NOTERMINAL] :  
2287 2417 5 BEGIN  
2288 2418 5 IF NOT .DBG$GB_DEF_OUT [OUT_LOG]  
2289 2419 5 THEN  
2290 2420 5     DBG$NOUT_INFO (DBG$_OUTPUTLOST);  
2291 2421 5  
2292 2422 5     DBG$GB_DEF_OUT [OUT_TERM] = FALSE;  
2293 2423 4 END;  
2294 2424 4  
2295 2425 4 [NOUN_LITERAL_VERIFY] :  
2296 2426 4     DBG$GB_DEF_OUT [OUT_VERIFY] = TRUE;  
2297 2427 4  
2298 2428 4 [NOUN_LITERAL_NOVERIFY] :  
2299 2429 4     DBG$GB_DEF_OUT [OUT_VERIFY] = FALSE;  
2300 2430 4  
2301 2431 4 TES;  
2302 2432 4  
2303 2433 4 ! Check for another object  
2304 2434 4 !  
2305 2435 4 IF .NOUN_NODE [DBG$L_NOUN_LINK] EQ LA 0  
2306 2436 4 THEN  
2307 2437 4     EXITLOOP;  
2308 2438 4  
2309 2439 4 ! Update the noun node  
2310 2440 4 !  
2311 2441 4 NOUN_NODE = .NOUN_NODE [DBG$L_NOUN_LINK];  
2312 2442 4  
2313 2443 3 END;  
2314 2444 3 ! End of loop over OUTPUT parameters  
2315 2445 2  
2316 2446 2 END;  
2317 2447 2  
2318 2448 2 ! Execute the SET RADIX command.  
2319 2449 2 !  
2320 2450 2 [SET_RADIX, SET_RADIX_OVERRIDE] :  
2321 2451 3 BEGIN  
2322 2452 3 LOCAL
```



```
2323 2453 3
2324 2454 3
2325 2455 3
2326 2456 3
2327 2457 3
2328 2458 3
2329 2459 3
2330 2460 3
2331 2461 3
2332 2462 3
2333 2463 3
2334 2464 3
2335 2465 3
2336 2466 3
2337 2467 3
2338 2468 3
2339 2469 3
2340 2470 3
2341 2471 3
2342 2472 3
2343 2473 3
2344 2474 3
2345 2475 3
2346 2476 3
2347 2477 3
2348 2478 3
2349 2479 3
2350 2480 3
2351 2481 3
2352 2482 3
2353 2483 3
2354 2484 3
2355 2485 3
2356 2486 3
2357 2487 3
2358 2488 3
2359 2489 4
2360 2490 4
2361 2491 4
2362 2492 4
2363 2493 4
2364 2494 4
2365 2495 4
2366 2496 4
2367 2497 4
2368 2498 4
2369 2499 4
2370 2500 4
2371 2501 4
2372 2502 4
2373 2503 4
2374 2504 3
2375 2505 3
2376 2506 3
2377 2507 3
2378 2508 3
2379 2509 3
```

```
    RADIX;
! Fill in the correct radix value.
CASE .NOUN_NODE [DBG$L_NOUN_VALUE]
    FROM MODE_LOWEST TO MODE_HIGHEST OF
    SET
        [MODE_BINARY] : RADIX = DBG$K_BINARY;
        [MODE_OCTAL]  : RADIX = DBG$K_OCTAL;
        [MODE_DECIMAL] : RADIX = DBG$K_DECIMAL;
        [MODE_DEFAULT] : RADIX = DBG$K_DEFAULT;
        [MODE_HEX]     : RADIX = DBG$K_HEX;
    [INRANGE, OVRANGE]:
        $DBG_ERROR('DBGNSET\DBG$NEXECUTE_SET unexpected radix');
    TES;
! The VALUE2 field contains:
!   0 if no /INPUT or /OUTPUT was specified (in this
!   case we set both radices)
!   1 if /INPUT was specified
!   2 if /OUTPUT was specified
! The VERB_COMPOSITE field tells us whether /OVERRIDE was
! specified.
! We use these pieces of information to tell us which
! of the three elements of the DBG$GB_RADIX byte vector to
! fill in.
IF .VERB_NODE[DBG$B_VERB_COMPOSITE] EQL SET_RADIX
THEN
    BEGIN
        IF .RADIX EQL DBG$K_DEFAULT
        THEN
            RADIX = DBG$NGET_TRANS_RADIX(.RADIX);

        IF .NOUN_NODE[DBG$L_NOUN_VALUE2] LEQ 1
        THEN
            DBG$GB_RADIX[DBG$B_RADIX_INPUT] = .RADIX;

        IF NOT .NOUN_NODE[DBG$L_NOUN_VALUE2]
        THEN
            DBG$GB_RADIX[DBG$B_RADIX_OUTPUT] = .RADIX;

        END
    ELSE IF .VERB_NODE[DBG$B_VERB_COMPOSITE] EQL SET_RADIX_OVERRIDE
    THEN
        DBG$GB_RADIX[DBG$B_RADIX_OUTPUT_OVER] = .RADIX
    ELSE
        $DBG_ERROR('DBGNSET\DBG$NEXECUTE_SET');
```



```

END;                                ! End of SET RADIX command

[set_scope] :
BEGIN
LOCAL
    SCOPE_LIST : REF VECTOR [,LONG],
    BAD_SCOPE,
    ERR_INDEX;

! Recover the scope vector
scope_list = .noun_node [dbg$l_noun_value];

! Set the scopes. 1 was filled into the adverb field if
! SET SCOPE/MODULE was specified; 0 otherwise.
dbg$rst_setscope (.scope_list, err_index,
                  .verb_node[dbg$l_verb_adverb_ptr]);

! Check for an error
IF .err_index NEQ 0
THEN
    BEGIN
        ! Translate the bad scope to a counted string
        dbg$npathdesc_to_cs (.scope_list [.err_index], bad_scope);

        ! Format an error message
        .message_vect = dbg$nmake_arg_vect (dbg$_badscope, 1, .bad_scope);

        RETURN sts$k_severe;
    END;
END;

[set_search] :
BEGIN
LITERAL
    noun_literal_all = 1,
    noun_literal_next = 2,
    noun_literal_string = 3,
    noun_literal_ident = 4;

! First set the search structure to the user defined level
dbg$set_search_lvl (user_def_search);

! Add the new user defined settings. Remember the first noun
! node has already been recovered.
WHILE TRUE DO
    BEGIN

```


2437 2567 4
2438 2568 4
2439 2569 4
2440 2570 4
2441 2571 4
2442 2572 4
2443 2573 5
2444 2574 5
2445 2575 4
2446 2576 4
2447 2577 4
2448 2578 5
2449 2579 5
2450 2580 4
2451 2581 4
2452 2582 4
2453 2583 5
2454 2584 5
2455 2585 4
2456 2586 4
2457 2587 4
2458 2588 5
2459 2589 5
2460 2590 4
2461 2591 4
2462 2592 4
2463 2593 4
2464 2594 4
2465 2595 4
2466 2596 4
2467 2597 4
2468 2598 4
2469 2599 4
2470 2600 4
2471 2601 4
2472 2602 4
2473 2603 4
2474 2604 3
2475 2605 3
2476 2606 2
2477 2607 2
2478 2608 2
2479 2609 3
2480 2610 3
2481 2611 3
2482 2612 3
2483 2613 3
2484 2614 3
2485 2615 2
2486 2616 2
2487 2617 2
2488 2618 2
2489 2619 3
2490 2620 3
2491 2621 2
2492 2622 2
2493 2623 2

```
CASE .noun_node [dbg$l_noun_value] FROM noun_literal_all
      TO noun_literal_ident
OF SET
  [noun_literal_all] : ! SET SEARCH ALL
  BEGIN
    dbg$gb_search_ptr[search_all] = TRUE;
  END;

  [noun_literal_next] : ! SET SEARCH NEXT
  BEGIN
    dbg$gb_search_ptr[search_all] = FALSE;
  END;

  [noun_literal_ident] : ! SET SEARCH IDENT
  BEGIN
    dbg$gb_search_ptr[search_ident] = TRUE;
  END;

  [noun_literal_string] : ! SET SEARCH STRING
  BEGIN
    dbg$gb_search_ptr[search_ident] = FALSE;
  END;

  TES;
  ! Now check for another switch. If none, exit the loop.
  IF .noun_node [dbg$l_noun_link] EQL 0
  THEN
    EXITLOOP;

  ! Update the noun node to the next object
  !
  noun_node = .noun_node [dbg$l_noun_link];
END;      ! End of loop

END;

[set_source] :      ! Set source directory search list
BEGIN
  dbg$src_set_source(
    .noun_node[dbg$l_adjective_ptr],
    .noun_node[dbg$l_noun_value]);
END;

[set_step] :      ! Set step values
BEGIN
  RETURN DBG$EVENT_SEMANTICS (.VERB_NODE, .MESSAGE_VECT);
END;
```



```
2494      ! Execute the SET TASK command.
2495      !
2496      [SET TASK]:
2497          DBG$NEXECUTE_SET_TASK(.VERB_NODE);
2498      !
2499      ! Execute the SET TERMINAL command.
2500      !
2501      [SET TERMINAL]:
2502          DBG$SCR_EXECUTE_SETTERM_CMD(.VERB_NODE);
2503      !
2504      ! Execute the SET TRACE command.
2505      !
2506      [SET TRACE]:
2507          DBG$EVENT_SEMANTICS(.VERB_NODE, .MESSAGE_VECT);
2508      !
2509      [set_type] :
2510          BEGIN
2511              LOCAL
2512                  LENGTH_NODE : REF dbg$noun_node;
2513              ! We always have a second noun node (containing the length).
2514              length_node = .noun_node [dbg$l_noun_link];
2515              dbg$gl_dflttyp = .noun_node [dbg$l_noun_value];
2516              dbg$gw_dfltleng = .length_node [dbg$l_noun_value];
2517              END;
2518      [set_type_override] :
2519          BEGIN
2520              LOCAL
2521                  LENGTH_NODE : REF dbg$noun_node;
2522              ! We always have a second noun node (containing the length).
2523              length_node = .noun_node [dbg$l_noun_link];
2524              dbg$gl_gbltyp = .noun_node [dbg$l_noun_value];
2525              dbg$gw_gbllength = .length_node [dbg$l_noun_value];
2526              END;
2527      ! Execute the SET WATCH command.
2528      !
2529      [set_watch]:
2530          DBG$EVENT_SEMANTICS (.VERB_NODE, .MESSAGE_VECT);
2531      !
2532      ! Execute the SET WINDOW command.
2533      !
2534      [SET WINDOW]:
2535          DBG$SCR_EXECUTE_SETWIND_CMD(.VERB_NODE);
2536      !
2537      !
2538      !
2539      !
2540      !
2541      !
2542      !
2543      !
2544      !
2545      !
2546      !
2547      !
2548      !
2549      !
2550      !
```



```
: 2551      2681  2      ! Any other command CASE index should never occur. Hence we signal
: 2552      2682  2      ! an internal DEBUG error in this case.
: 2553      2683  2
: 2554      2684  2
: 2555      2685  2      [INRANGE, OUTRANGE]:
: 2556      2686  2      $DBG_ERROR('DBGNSET\NEXECUTE_SET');
: 2557      2687  2
: 2558      2688  2      TES;
: 2559      2689  2
: 2560      2690  2      ! The command has successfully been executed. Now return.
: 2561      2691  2      RETURN STS$K_SUCCESS;
: 2562      2692  2
: 2563      2693  2
: 2564      2694  1      END;
```

```
                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0
45  4E  24  47  42  44  5C  54  45  53  4E  47  42  44  29  00309 P.AEG: .ASCII  \)DBGNSET\<92>\DBG$NEXECUTE_SET unexpect\
78  65  6E  75  20  54  45  53  5F  45  54  55  43  45  58  00318
                                74  63  65  70  00327
                                0032B
45  4E  24  47  42  44  5C  78  69  64  61  72  20  64  65  00333 P.AEH: .ASCII  \ed radix\
55  43  45  58  45  4E  5C  54  45  53  5F  45  54  55  43  45  58  00342 P.AEI: .ASCII  <24>\DBGNSET\<92>\DBG$NEXECUTE_SET\
                                54  45  53  5F  45  54  0034C P.AEI: .ASCII  <20>\DBGNSET\<92>\NEXECUTE_SET\
                                0035B

                                .EXTRN  LIB$SPAWN, SYSS$WAITFR
                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0
                                OFFC 00000
                                .ENTRY  DBG$NEXECUTE_SET, Save R2,R3,R4,R5,R6,R7,-
                                R8,R9,R10,R11
                                MOVAB   DBG$NGET_TRANS_RADIX, R11
                                MOVAB   DBG$GB_MOD_PTR, R10
                                MOVAB   LIB$SIGNAL, R9
                                MOVAB   DBG$GL_LOGFAB+52, R8
                                MOVAB   DBG$GB_RADIX, R7
                                MOVAB   DBG$GB_DEF_OUT, R6
                                SUBL2   #24, SP
                                MOVL    VERB_NODE, R2
                                MOVL    8(R2), NOUN_NODE
                                CASEB   1(R2), #1, #29
                                .WORD   98$-1$,-
                                2$-1$,-
                                8$-1$,-
                                11$-1$,-
                                13$-1$,-
                                18$-1$,-
                                37$-1$,-
                                37$-1$,-
                                43$-1$,-
                                80$-1$,-
                                93$-1$,-
                                96$-1$,-
                                97$-1$,-

                                1997
                                2055
                                2060
```

00BB	1D	003C	047D	0003C	1\$:
01AF	0083	00FA	00C8	00044	
0455	0432	03AC	0241	0004C	
003C	003C	047D	0469	00054	
00DD	00EE	0424	047D	0005C	
0076	0061	0045	03EB	00064	
0321	0321	048B	044A	0006C	
		043F	008C	00074	

						98\$-1\$,-			
						2\$-1\$,-			
						2\$-1\$,-			
						98\$-1\$,-			
						92\$-1\$,-			
						16\$-1\$,-			
						14\$-1\$,-			
						84\$-1\$,-			
						3\$-1\$,-			
						6\$-1\$,-			
						7\$-1\$,-			
						95\$-1\$,-			
						99\$-1\$,-			
						63\$-1\$,-			
						63\$-1\$,-			
						9\$-1\$,-			
						94\$-1\$			
		00000000'	EF	9F	00078	2\$:	PUSHAB	P.AEI	2685
			0359	31	0007E		BRW	77\$	
		08	A2	D5	00081	3\$:	TSTL	8(R2)	2075
			7C	13	00084		BEQL	12\$	
	53	08	A2	D0	00086		MOVL	8(R2), NOUN_NODE	2078
00	00000000'	EF	63	E2	0008A	4\$:	BBSS	(NOUN_NODE), DBG\$GL_DEVELOPER, 5\$	2081
		08	A3	D5	00092	5\$:	TSTL	8(NOUN_NODE)	2082
			6B	13	00095		BEQL	12\$	
	53	08	A3	D0	00097		MOVL	8(NOUN_NODE), NOUN_NODE	2083
			ED	11	0009B		BRB	4\$	2079
			01	DD	0009D	6\$:	PUSHL	#1	2095
00000000G	00		01	FB	0009F		CALLS	#1, DBG\$SET_DEFINE_LVL	
50	00000000G		00	D0	000A6		MOVL	DBG\$GB_DEFINE_PTR, R0	2096
60			63	90	000AD		MOVB	(NOUN_NODE), (R0)	
			76	11	000B0		BRB	15\$	2060
			01	DD	000B2	7\$:	PUSHL	#1	2103
			52	DD	000B4		PUSHL	R2	
00000000G	00		02	FB	000B6		CALLS	#2, DBG\$SCR_EXECUTE_DISPLAY_CMD	
			74	11	000BD		BRB	17\$	
		00000000G	00	94	000BF	8\$:	CLRB	DBG\$GB_RESIGNAL	2110
			03A6	31	000C5		BRW	93\$	2111
	54	04	A2	D0	000C8	9\$:	MOVL	4(R2), OUTPUT_LOG	2125
			63	DD	000CC		PUSHL	(NOUN_NODE)	2131
		00000000G	00	9F	000CE		PUSHAB	DBG\$GC_KEY_TABLE_ID	2130
00000000G	00		02	FB	000D4		CALLS	#2, SMG\$SET_DEFAULT_STATE	
09			50	E8	000DB		BLBS	SET STATUS, -10\$	2132
		00028150	8F	DD	000DE		PUSHL	#164176	
69			01	FB	000E4		CALLS	#1, LIB\$SIGNAL	
49			54	E9	000E7	10\$:	BLBC	OUTPUT_LOG, 17\$	2133
			63	DD	000EA		PUSHL	(NOUN_NODE)	2135
			01	DD	000EC		PUSHL	#1	
		000280C3	8F	DD	000EE		PUSHL	#164035	
			02EB	31	000F4		BRW	78\$	
			63	DD	000F7	11\$:	PUSHL	(NOUN_NODE)	2141
			7E	D4	000F9		CLRL	-(SP)	
00000000G	00		02	FB	000FB		CALLS	#2, DBG\$SET_LANG	
			2F	11	00102	12\$:	BRB	17\$	2060
00000000G	00		63	D0	00104	13\$:	MOVL	(NOUN_NODE), DBG\$GL_LOG_BUF	2146
		08	AC	DD	0010B		PUSHL	MESSAGE_VECT	2147
0000V	CF		01	FB	0010E		CALLS	#1, DBG\$NSET_LOG	

		1D		50	E8	00113		BLBS	R0, 17\$		
				030A	31	00116		BRW	83\$		2149
		00000000G	00	63	D0	00119	14\$:	MOVL	(NOUN_NODE), DBG\$SRC_LEFT_MARGIN		2154
		00000000G	00	OC	A3	00120		MOVL	12(NOUN_NODE), DBG\$SRC_RIGHT_MARGIN		2155
				09	11	00128	15\$:	BRB	17\$		2060
				63	DD	0012A	16\$:	PUSHL	(NOUN_NODE)		2160
		00000000G	00	01	FB	0012C		CALLS	#1, DBG\$SRC_SET_MAX_FILES		
				039A	31	00133	17\$:	BRW	100\$		2060
				01	DD	00136	18\$:	PUSHL	#1		2172
		00000000G	00	01	FB	00138		CALLS	#1, DBG\$SET_MOD_LVL		
				53	D5	0013F	19\$:	TSTL	NOUN_NODE		2177
				F0	13	00141		BEQL	17\$		
				63	CF	00143		CASEL	(NOUN_NODE), #1, #12		2179
0042	OC		01	001A		00147	20\$:	.WORD	21\$-20\$,-		
006B	0028		0021	0049		0014F			22\$-20\$,-		
005A	002F		0052	006F		00157			23\$-20\$,-		
	0086		007A	0063		0015F			25\$-20\$,-		
									26\$-20\$,-		
									27\$-20\$,-		
									24\$-20\$,-		
									30\$-20\$,-		
									31\$-20\$,-		
									33\$-20\$,-		
									34\$-20\$,-		
									28\$-20\$,-		
									29\$-20\$		
		67	0202	8F	B0	00161	21\$:	MOVW	#514, DBG\$GB_RADIX		2185
				7C	11	00166		BRB	36\$		2179
		67	0808	8F	B0	00168	22\$:	MOVW	#2056, DBG\$GB_RADIX		2191
				75	11	0016D		BRB	36\$		2179
		67	0A0A	8F	B0	0016F	23\$:	MOVW	#2570, DBG\$GB_RADIX		2197
				6E	11	00174		BRB	36\$		2179
				01	DD	00176	24\$:	PUSHL	#1		2204
		6B		01	FB	00178		CALLS	#1, DBG\$NGET_TRANS_RADIX		
		67		50	90	0017B		MOVB	R0, DBG\$GB_RADIX		
				01	DD	0017E		PUSHL	#1		2206
		6B		01	FB	00180		CALLS	#1, DBG\$NGET_TRANS_RADIX		
01		A7		50	90	00183		MOVB	R0, DBG\$GB_RADIX+1		
				5B	11	00187		BRB	36\$		2179
		67	1010	8F	B0	00189	25\$:	MOVW	#4112, DBG\$GB_RADIX		2211
				54	11	0018E		BRB	36\$		2179
		50		6A	D0	00190	26\$:	MOVL	DBG\$GB_MOD_PTR, R0		2216
02		A0		01	90	00193		MOVB	#1, 2(R0)		
				4B	11	00197		BRB	36\$		
		50		6A	D0	00199	27\$:	MOVL	DBG\$GB_MOD_PTR, R0		2219
			02	A0	94	0019C		CLRB	2(R0)		
				43	11	0019F		BRB	36\$		
		50		6A	D0	001A1	28\$:	MOVL	DBG\$GB_MOD_PTR, R0		2222
09		A0		01	90	001A4		MOVB	#1, 9(R0)		
				3A	11	001A8		BRB	36\$		
		50		6A	D0	001AA	29\$:	MOVL	DBG\$GB_MOD_PTR, R0		2225
			09	A0	94	001AD		CLRB	9(R0)		
				32	11	001B0		BRB	36\$		
				01	DD	001B2	30\$:	PUSHL	#1		2228
				02	11	001B4		BRB	32\$		
				7E	D4	001B6	31\$:	CLRL	-(SP)		2231
		00000000G	00	01	FB	001B8	32\$:	CALLS	#1, DBG\$SCR_SCREEN_MODE		

00000000G	00	23	11	001BF	BRB	36\$				
	6E	01	90	001C1	33\$:	MOVB	#1, DBG\$GB_KEYPAD_INPUT	2235		
		01	D0	001C8		MOVL	#1, (SP)	2236		
		08	11	001CB		BRB	35\$			
		00000000G	00	94	001CD	34\$:	CLRB	DBG\$GB_KEYPAD_INPUT		
			6E	D4	001D3		CLRL	(SP)		
			5E	DD	001D5	35\$:	PUSHL	SP		
		00000000G	00	9F	001D7		PUSHAB	DBG\$GL_KEYBOARD_ID		
			02	FB	001DD		CALLS	#2, SMG\$SET_KEYPAD_MODE		
		00000000G	00	A3	D0	001E4	36\$:	MOVL	8(NOUN_NODE), NOUN_NODE	
			54	31	001E8		BRW	19\$		
			55	D4	001EB	37\$:	CLRL	STATUS		
			A2	E9	001ED		BLBC	4(R2), 38\$		
	08	2F	04	01	D0	001F1		MOVL	#1, FLAGS	
		AE	04	AE	D4	001F5		CLRL	EVNT_FLAG	
			04	AE	9F	001F8		PUSHAB	EVNT_FLAG	
			7E	7C	001FB		CLRQ	-(SP)		
			7E	D4	001FD		CLRL	-(SP)		
			18	AE	9F	001FF		PUSHAB	FLAGS	
			7E	7C	00202		CLRQ	-(SP)		
			7E	D4	00204		CLRL	-(SP)		
		00000000G	00	08	FB	00206		CALLS	#8, LIB\$SPAWN	
			55	50	D0	0020D		MOVL	R0, STATUS	
			0D	55	E8	00210		BLBS	STATUS, 38\$	
				55	DD	00213		PUSHL	STATUS	
				7E	D4	00215		CLRL	-(SP)	
			000286DB	8F	DD	00217		PUSHL	#165595	
			69	03	FB	0021D		CALLS	#3, LIB\$SIGNAL	
	54	04	A2	02	C7	00220	38\$:	DIVL3	#2, 4(R2), ALLOCATE_FLAG	
			08	A2	91	00225		CMPB	1(R2), #8	
				0D	12	00229		BNEQ	39\$	
				54	DD	0022B		PUSHL	ALLOCATE_FLAG	
				7E	7C	0022D		CLRQ	-(SP)	
			00000000G	00	03	FB	0022F		CALLS	#3, DBG\$RST_SETMOD
				2C	11	00236		BRB	41\$	
				53	D5	00238	39\$:	TSTL	NOUN_NODE	
				28	13	0023A		BEQL	41\$	
			52	63	D0	0023C		MOVL	(NOUN_NODE), NAME_BUFF	
				54	DD	0023F		PUSHL	ALLOCATE_FLAG	
			7E	62	9A	00241		MOVZBL	(NAME_BUFF), -(SP)	
				A2	9F	00244		PUSHAB	1(NAME_BUFF)	
			01	03	FB	00247		CALLS	#3, DBG\$RST_SETMOD	
			00000000G	00	50	E8	0024E		BLBS	R0, 40\$
				52	DD	00251		PUSHL	NAME_BUFF	
				01	DD	00253		PUSHL	#1	
			000281E8	8F	DD	00255		PUSHL	#164328	
				01BA	31	0025B		BRW	81\$	
			53	A3	D0	0025E	40\$:	MOVL	8(NOUN_NODE), NOUN_NODE	
				D4	11	00262		BRB	39\$	
			13	55	E9	00264	41\$:	BLBC	STATUS, 42\$	
				8F	DD	00267		PUSHL	#165603	
			69	01	FB	0026D		CALLS	#1, LIB\$SIGNAL	
				AE	DD	00270		PUSHL	EVNT_FLAG	
			04	01	FB	00273		CALLS	#1, SYSS\$WAITFR	
			00000000G	00	0253	31	0027A	42\$:	BRW	100\$
				63	CF	0027D	43\$:	CASEL	(NOUN_NODE), #1, #7	
				0010		00281	44\$:	.WORD	45\$-47\$,-	

00A1

07
009101
007C

00CD

00C7

00B2

00AC

00289

						51\$-44\$,-				
						53\$-44\$,-				
						55\$-44\$,-				
						57\$-44\$,-				
						58\$-44\$,-				
						60\$-44\$,-				
						61\$-44\$,-				
						B TYPE	2374			
0000V	CF		OC	AE	9F	00291 45\$:	PUSHAB	CALLS	#T, DBG\$NSETUP_LOG	
	58			01	FB	00294	BLBS	STATUS, 50\$		2375
				50	E8	00299	PUSHL	#2		2377
00000000G	00			02	DD	0029C	CALLS	#1, DBG\$GET_TEMP MEM		
		00000000G		01	FB	0029E	TSTL	DBG\$GL_LOG_BUF		
				00	D5	002A5	BEQL	46\$		
	60			0A	13	002AB	MOVZBW	DBG\$GL_LOGFAB+52, (MSG_DESC)		
04	A0		F8	68	9B	002AD	MOVL	DBG\$GL_LOGFAB+44, 4(MSG_DESC)		
				A8	D0	002B0	BRB	47\$		
	60		01	09	11	002B5	MOVZBW	DBG\$GL_LOGFAB+53, (MSG_DESC)		
04	A0		FC	A8	9B	002B7 46\$:	MOVL	DBG\$GL_LOGFAB+48, 4(MSG_DESC)		
	01		OC	A8	D0	002BB	CMPL	B TYPE, #1		
				AE	D1	002C0 47\$:	BNEQ	48\$		
				0A	12	002C4	MOVL	DBG\$GL_LOGFAB+8, STS		
	52		D4	A8	D0	002C6	MOVL	DBG\$GL_LOGFAB+12, STV		
	51		D8	A8	D0	002CA	BRB	49\$		
				0E	11	002CE	MOVL	DBG\$GL_LOGRAB+8, STS		
52	00000000G			00	D0	002D0 48\$:	MOVL	DBG\$GL_LOGRAB+12, STV		
51	00000000G			00	D0	002D7	PUSHL	STV		
				51	DD	002DE 49\$:	PUSHR	#^M<R0,R2>		
				05	BB	002E0	PUSHL	#1		
				01	DD	002E2	PUSHL	#135328		
00000000G	00	000210A0		8F	DD	002E4	CALLS	#5, DBG\$NMAKE_ARG_VECT		
				05	FB	002EA	BRW	82\$		
	66			01	90	002F1 50\$:	MOVB	#1, DBG\$GB_DEF_OUT		2379
	56		03	A6	E9	002F7	BLBC	DBG\$GB_DEF_OUT+3, 62\$		2380
				1C	11	002FB	BRB	54\$		2382
	0D		01	A6	E8	002FD 51\$:	BLBS	DBG\$GB_DEF_OUT+1, 52\$		2388
		00028083		8F	DD	00301	PUSHL	#16397T		2390
00000000G	00			01	FB	00307	CALLS	#1, DBG\$NOUT_INFO		
				66	94	0030E 52\$:	CLRB	DBG\$GB_DEF_OUT		2392
				13	11	00310	BRB	56\$		2393
03	A6			01	90	00312 53\$:	MOVB	#1, DBG\$GB_DEF_OUT+3		2398
	38			66	E9	00316	BLBC	DBG\$GB_DEF_OUT, 62\$		2399
00000000G	00			01	D0	00319 54\$:	MOVL	#1, DBG\$GL_SCREEN_LOG		2401
				2F	11	00320	BRB	62\$		2364
		03		A6	94	00322 55\$:	CLRB	DBG\$GB_DEF_OUT+3		2407
		00000000G		00	D4	00325 56\$:	CLRL	DBG\$GL_SCREEN_LOG		2408
				24	11	0032B	BRB	62\$		2364
01	A6			01	90	0032D 57\$:	MOVB	#1, DBG\$GB_DEF_OUT+1		2413
				1E	11	00331	BRB	62\$		2364
	0D			66	E8	00333 58\$:	BLBS	DBG\$GB_DEF_OUT, 59\$		2418
		00028083		8F	DD	00336	PUSHL	#16397T		2420
00000000G	00			01	FB	0033C	CALLS	#1, DBG\$NOUT_INFO		
			01	A6	94	00343 59\$:	CLRB	DBG\$GB_DEF_OUT+1		2422
				09	11	00346	BRB	62\$		2364
02	A6			01	90	00348 60\$:	MOVB	#1, DBG\$GB_DEF_OUT+2		2426
				03	11	0034C	BRB	62\$		
			02	A6	94	0034E 61\$:	CLRB	DBG\$GB_DEF_OUT+2		2429

0041
001A
001A

OC
0037
003C
001A

01
0032
001A
001A

53

08	A3	D5	00351	62\$:	TSTL	8(NOUN_NODE)
	7C	13	00354		BEQL	75\$
08	A3	D0	00356		MOVL	8(NOUN_NODE), NOUN_NODE
	FF20	31	0035A		BRW	43\$
	63	CF	0035D	63\$:	CASEL	(NOUN_NODE), #1, #12
	002D		00361	64\$:	.WORD	66\$-64\$,-
	001A		00369			67\$-64\$,-
	001A		00371			68\$-64\$,-
	001A		00379			70\$-64\$,-

2435
2441
2362
2457

	00000000'	EF	9F	0037B	65\$:	PUSHAB	P.AEG
		01	DD	00381		PUSHL	#1
	00028362	8F	DD	00383		PUSHL	#164706
69		03	FB	00389		CALLS	#3, LIB\$SIGNAL
		17	11	0038C		BRB	71\$
50		02	D0	0038E	66\$:	MOVL	#2, RADIX
		12	11	00391		BRB	71\$
50		08	D0	00393	67\$:	MOVL	#8, RADIX
		0D	11	00396		BRB	71\$
50		0A	D0	00398	68\$:	MOVL	#10, RADIX
		08	11	0039B		BRB	71\$
50		01	D0	0039D	69\$:	MOVL	#1, RADIX
		03	11	003A0		BRB	71\$
50		10	D0	003A2	70\$:	MOVL	#16, RADIX
1B	01	A2	91	003A5	71\$:	CMPB	1(R2), #27
		1D	12	003A9		BNEQ	74\$
01		50	D1	003AB		CMPL	RADIX, #1
		05	12	003AE		BNEQ	72\$
		50	DD	003B0		PUSHL	RADIX
68		01	FB	003B2		CALLS	#1, DBG\$NGET TRANS_RADIX
01	OC	A3	D1	003B5	72\$:	CMPL	12(NOUN_NODE), #1
		03	14	003B9		BGTR	73\$
67		50	90	003BB		MOVB	RADIX, DBG\$GB_RADIX
23	OC	A3	E8	003BE	73\$:	BLBS	12(NOUN_NODE), 79\$
01		50	90	003C2		MOVB	RADIX, DBG\$GB_RADIX+1
		1D	11	003C6		BRB	79\$
1C	01	A2	91	003C8	74\$:	CMPB	1(R2), #28
		06	12	003CC		BNEQ	76\$
02	A7	50	90	003CE		MOVB	RADIX, DBG\$GB_RADIX+2
		11	11	003D2	75\$:	BRB	79\$
	00000000'	EF	9F	003D4	76\$:	PUSHAB	P.AEH
		01	DD	003DA	77\$:	PUSHL	#1
	00028362	8F	DD	003DC		PUSHL	#164706
69		03	FB	003E2	78\$:	CALLS	#3, LIB\$SIGNAL
		00E8	31	003E5	79\$:	BRW	100\$
54		63	D0	003E8	80\$:	MOVL	(NOUN_NODE), SCOPE_LIST
	04	A2	DD	003EB		PUSHL	4(R2)
	14	AE	9F	003EE		PUSHAB	ERR_INDEX
		54	DD	003F1		PUSHL	SCOPE_LIST

2472
2461
2463
2465
2467
2469
2487
2490
2492
2494
2496
2498
2500
2487
2504
2506
2509
2060
2523
2529
2528

00000000G	00	03	FB	003F3	CALLS	#3, DBG\$RST_SETSCOPE	2533
	50	10	AE	D0 003FA	MOVL	ERR_INDEX, R0	
			E5	13 003FE	BEQL	79\$	
		14	AE	9F 00400	PUSHAB	BAD_SCOPE	2539
			64	40 DD 00403	PUSHL	(SCOPE_LIST)[R0]	
00000000G	00		02	FB 00406	CALLS	#2, DBG\$NPATHDESC_TO_CS	
		14	AE	DD 0040D	PUSHL	BAD_SCOPE	2543
			01	DD 00410	PUSHL	#1	
		000281D8	8F	DD 00412	PUSHL	#164312	
00000000G	00		03	FB 00418	CALLS	#3, DBG\$NMAKE_ARG_VECT	
	08		50	D0 0041F	MOVL	R0, @MESSAGE_VECT	
	50		04	D0 00423	MOVL	#4, R0	2545
				04 00426	RET		
			01	DD 00427	PUSHL	#1	2560
00000000G	00		01	FB 00429	CALLS	#1, DBG\$SET_SEARCH_LVL	
	50	00000000G	00	D0 00430	MOVL	DBG\$GB_SEARCH_PTR, R0	2574
	01		63	CF 00437	CASEL	(NOUN_NODE), #1, #3	2568
0011	0017	000D	0008	0043B	.WORD	87\$-86\$,-	
						88\$-86\$,-	
						90\$-86\$,-	
						89\$-86\$	
	60		01	90 00443	MOVB	#1, (R0)	2574
			0D	11 00446	BRB	91\$	2568
			60	94 00448	CLRB	(R0)	2579
			09	11 0044A	BRB	91\$	2568
	01	A0	01	90 0044C	MOVB	#1, 1(R0)	2584
			03	11 00450	BRB	91\$	2568
			01	A0 94 00452	CLRB	1(R0)	2589
			08	A3 D5 00455	TSTL	8(NOUN_NODE)	2596
			76	13 00458	BEQL	100\$	
	53		08	A3 D0 0045A	MOVL	8(NOUN_NODE), NOUN_NODE	2602
			D7	11 0045E	BRB	85\$	2565
			63	DD 00460	PUSHL	(NOUN_NODE)	2613
			04	A3 DD 00462	PUSHL	4(NOUN_NODE)	2612
00000000G	00		02	FB 00465	CALLS	#2, DBG\$SRC_SET_SOURCE	
			62	11 0046C	BRB	100\$	2060
			08	AC DD 0046E	PUSHL	MESSAGE_VECT	2620
			52	DD 00471	PUSHL	R2	
00000000G	00		02	FB 00473	CALLS	#2, DBG\$EVENT_SEMANTICS	
				04 0047A	RET		
			52	DD 0047B	PUSHL	R2	2627
00000000G	00		01	FB 0047D	CALLS	#1, DBG\$NEXECUTE_SET_TASK	
			4A	11 00484	BRB	100\$	
			52	DD 00486	PUSHL	R2	2633
00000000G	00		01	FB 00488	CALLS	#1, DBG\$SCR_EXECUTE_SETTERM_CMD	
			3F	11 0048F	BRB	100\$	
	50		08	A3 D0 00491	MOVL	8(NOUN_NODE), LENGTH_NODE	2649
00000000G	00		63	D0 00495	MOVL	(NOUN_NODE), DBG\$GL_DFLTYP	2651
00000000G	00		60	B0 0049C	MOVW	(LENGTH_NODE), DBG\$GW_DFLTLENG	2652
			2B	11 004A3	BRB	100\$	2060
	50		08	A3 D0 004A5	MOVL	8(NOUN_NODE), LENGTH_NODE	2662
00000000G	00		63	D0 004A9	MOVL	(NOUN_NODE), DBG\$GL_GBLTYP	2664
00000000G	00		60	B0 004B0	MOVW	(LENGTH_NODE), DBG\$GW_GBLLENGTH	2665
			17	11 004B7	BRB	100\$	2060
			08	AC DD 004B9	PUSHL	MESSAGE_VECT	2672
			52	DD 004BC	PUSHL	R2	
00000000G	00		02	FB 004BE	CALLS	#2, DBG\$EVENT_SEMANTICS	

DBGNSET
V04-000

H
16-Sep-1984 01:58:19
14-Sep-1984 12:17:21

VAX-11 Bliss-32 V4.0-742
[DEBUG.SRC]DBGNSET.B32;1

Page 81
(4)

00000000G 00
50

09 11 004C5
52 DD 004C7 99\$:
01 FB 004C9
01 D0 004D0 100\$:
04 004D3

BRB 100\$
PUSHL R2
CALLS #1, DBG\$SCR_EXECUTE_SETWIND_CMD
MOVL #1, R0
RET

: 2678
:
:
:
: 2692
:
: 2694

; Routine Size: 1236 bytes, Routine Base: DBG\$CODE + 10E5


```
: 2566      2695 1 GLOBAL ROUTINE DBG$NSET_LOG (MESSAGE_VECT) =
: 2567      2696 1 ++
: 2568      2697 1 FUNCTIONAL DESCRIPTION:
: 2569      2698 1     This routine closes any log file that is currently open and opens
: 2570      2699 1     the file specified by the user in SET LOG "filespec"
: 2571      2700 1
: 2572      2701 1 FORMAL PARAMETERS:
: 2573      2702 1
: 2574      2703 1     message_vect    - The address of a longword to contain the address of a
: 2575      2704 1                      message argument vector
: 2576      2705 1
: 2577      2706 1 IMPLICIT INPUTS:
: 2578      2707 1     The RMS-set IFI (internal file identifier field) bit is used to
: 2579      2708 1     determine if a previous log file is still open.
: 2580      2709 1
: 2581      2710 1     dbg$gl_log_buf _    pointer to log filespec buffer
: 2582      2711 1
: 2583      2712 1 IMPLICIT OUTPUTS:
: 2584      2713 1     none
: 2585      2714 1
: 2586      2715 1 ROUTINE VALUE:
: 2587      2716 1
: 2588      2717 1     sts$k_severe (4) -    the log could not be set to the specified filespec
: 2589      2718 1
: 2590      2719 1     sts$k_success (1) -   the log file was set to the user specified filespec
: 2591      2720 1
: 2592      2721 1 SIDE EFFECTS:
: 2593      2722 1     Any previously open log file is closed and a new log file is opened.
: 2594      2723 1     Logging is inhibited for the duration of this routine.
: 2595      2724 1 --
: 2596      2725 1
: 2597      2726 2     BEGIN
: 2598      2727 2
: 2599      2728 2     LOCAL
: 2600      2729 2
: 2601      2730 2         STATUS,
: 2602      2731 2         OPENED_LOG,
: 2603      2732 2         FNS : BYTE,
: 2604      2733 2         FNA,
: 2605      2734 2         OLD_NAM_PTR,
: 2606      2735 2         B TYPE,
: 2607      2736 2         TEMP_NAM : REF $NAM DECL,
: 2608      2737 2         TEMP_FSE : REF VECTOR [,BYTE],
: 2609      2738 2         TEMP_FSR : REF VECTOR [,BYTE],
: 2610      2739 2         LOG_TEMP : BYTE;
: 2611      2740 2
: 2612      2741 2     opened_log = FALSE ;
: 2613      2742 2
: 2614      2743 2     ! inhibit logging until we can redirect the log file
: 2615      2744 2
: 2616      2745 2     log_temp = .dbg$gb_def_out [out_log];
: 2617      2746 2     dbg$gb_def_out [out_log] = FALSE;
: 2618      2747 2
: 2619      2748 2     ! see if a log file has been previously opened. If so, close it,
: 2620      2749 2     ! since it must be CLOSED before we can do the new $PARSE.
: 2621      2750 2
: 2622      2751 2     IF .dbg$gl_logfab [fab$w_ifi] NEQ 0
:                THEN
```



```

: 2623      2752      3      BEGIN
: 2624      2753
: 2625      2754      opened_log = TRUE;
: 2626      2755      fna = .dbg$gl_logfab [fab$l_fna];
: 2627      2756      fns = .dbg$gl_logfab [fab$b_fns];
: 2628      2757
: 2629      2758      status = %CLOSE (FAB = dbg$gl_logfab);
: 2630      2759      IF NOT .status
: 2631      2760      THEN
: 2632      2761          BEGIN
: 2633      2762              LOCAL
: 2634      2763                  MSG_DESC : REF dbg$stg_desc;
: 2635      2764
: 2636      2765                  msg_desc = dbg$get_tempmem(2);
: 2637      2766                  build_error_desc (msg_desc);
: 2638      2767                  .message_vect = dbg$nmake_arg_vect
: 2639      2768                      (shr$_closeout+dbg_fac_code,
: 2640      2769                      1, .msg_desc,
: 2641      2770                      .dbg$gl_logfab[fab$l_sts],
: 2642      2771                      .dbg$gl_logfab[fab$l_stv]);
: 2643      2772
: 2644      2773
: 2645      2774                  RETURN sts$k_severe;
: 2646      2775
: 2647      2776              END;
: 2648      2777
: 2649      2778      ! get new NAM block, preserve the original
: 2650      2779
: 2651      2780      temp_nam = dbg$get_memory ((nam$c_bln+3) / %UPVAL);
: 2652      2781      $nam_init (nam = .temp_nam);
: 2653      2782      temp_fse = dbg$get_memory ((nam$c_maxrss+3)/%UPVAL);
: 2654      2783      temp_fsr = dbg$get_memory ((nam$c_maxrss+3)/%UPVAL);
: 2655      2784      temp_nam [nam$l_esd] = .temp_fse;
: 2656      2785      temp_nam [nam$l_rsa] = .temp_fsr;
: 2657      2786      temp_nam [nam$b_ess] = nam$c_maxrss;
: 2658      2787      temp_nam [nam$b_rss] = nam$c_maxrss;
: 2659      2788      dbg$gl_logfab [fab$l_nam] = .temp_nam;
: 2660      2789      old_nam_ptr = .dbg$gl_lognam;
: 2661      2790      dbg$gl_lognam = .temp_nam;
: 2662      2791
: 2663      2792      END;
: 2664      2793
: 2665      2794      ! reset the appropriate FAB file process options
: 2666      2795
: 2667      2796      dbg$gl_logfab [fab$v_cif] = 0;
: 2668      2797      dbg$gl_logfab [fab$v_mxv] = 1;
: 2669      2798      dbg$gl_logfab [fab$v_nam] = 0;
: 2670      2799
: 2671      2800      ! Set up file name
: 2672      2801
: 2673      2802      IF .dbg$gl_log_buf NEQ 0
: 2674      2803      THEN
: 2675      2804          BEGIN
: 2676      2805              MAP
: 2677      2806                  DBG$GL_LOG_BUF : REF VECTOR [,BYTE];
: 2678      2807
: 2679      2808
```



```
2680 2809 3      dbg$gl_logfab [fab$l_fna] = .dbg$gl_log_buf + 1; ! filename addr starts in 2nd byte
2681 2810 3      dbg$gl_logfab [fab$b_fns] = .dbg$gl_log_buf [0]; ! 1st byte is count
2682 2811 3
2683 2812 3      ! Parse the filespec to see if an explicit version number was
2684 2813 3      ! given. If so set the CIF bit, otherwise we must maximize
2685 2814 3      ! the version number.
2686 2815 3
2687 2816 3      status = $PARSE (FAB = dbg$gl_logfab);
2688 2817 3      IF NOT .status
2689 2818 3      THEN
2690 2819 4          BEGIN
2691 2820 4              LOCAL
2692 2821 4                  STS,
2693 2822 4                  STV,
2694 2823 4                  MSG_DESC : REF dbg$stg_desc;
2695 2824 4
2696 2825 4          msg_desc = dbg$get_tempmem (2);
2697 2826 4
2698 2827 4          ! save these because restore_nam will reset them
2699 2828 4
2700 2829 4          sts = .dbg$gl_logfab [fab$l_sts];
2701 2830 4          stv = .dbg$gl_logfab [fab$l_stv];
2702 2831 4          build_error_desc (msg_desc);
2703 2832 4
2704 2833 4          IF .opened_log
2705 2834 4          THEN
2706 2835 5              restore_nam
2707 2836 4          ELSE
2708 2837 5              BEGIN
2709 2838 5                  dbg$gl_logfab [fab$l_fna] = 0;
2710 2839 5                  dbg$gl_logfab [fab$b_fns] = 0;
2711 2840 5                  dbg$gl_log_buf = 0;
2712 2841 4                  END;
2713 2842 4
2714 2843 4          .message_vect = dbg$nmake_arg_vect
2715 2844 4                      (shr$syntax + dbg_fac_code,
2716 2845 4                      1, .msg_desc, .sts, .stv);
2717 2846 4
2718 2847 4          RETURN sts$k_severe;
2719 2848 4
2720 2849 3      END;
2721 2850 3
2722 2851 3      IF .dbg$gl_lognam [nam$v_exp_ver]
2723 2852 3      THEN
2724 2853 4          BEGIN
2725 2854 4              dbg$gl_logfab [fab$v_cif] = 1;
2726 2855 4              dbg$gl_logfab [fab$v_mxv] = 0;
2727 2856 4              END;
2728 2857 3
2729 2858 3      dbg$gl_logfab [fab$v_nam] = 1;
2730 2859 2      END;
2731 2860 2
2732 2861 2      ! now open the new log file
2733 2862 2
2734 2863 2      status = dbg$nsetup_log (b_type);
2735 2864 2      IF NOT .status
2736 2865 2      THEN
```



```

: 2737      2866      BEGIN
: 2738      2867      LOCAL
: 2739      2868          MSG_DESC : REF dbg$stg_desc,
: 2740      2869          STS;
: 2741      2870          STV;
: 2742      2871
: 2743      2872      msg_desc = dbg$get_tempmem (2);
: 2744      2873      build_error_desc (msg_desc);
: 2745      2874      IF .b_type EQL 1
: 2746      2875      THEN
: 2747      2876          BEGIN
: 2748      2877              sts = .dbg$gl_logfab [fab$l_sts];
: 2749      2878              stv = .dbg$gl_logfab [fab$l_stv];
: 2750      2879          END
: 2751      2880      ELSE
: 2752      2881          BEGIN
: 2753      2882              sts = .dbg$gl_lograb [rab$l_sts];
: 2754      2883              stv = .dbg$gl_lograb [rab$l_stv];
: 2755      2884          END;
: 2756      2885
: 2757      2886      IF .opened_log
: 2758      2887      THEN
: 2759      2888          restore_nam
: 2760      2889      ELSE
: 2761      2890          BEGIN
: 2762      2891              dbg$gl_logfab [fab$l_fna] = 0;
: 2763      2892              dbg$gl_logfab [fab$b_fns] = 0;
: 2764      2893              dbg$gl_logbuf = 0;
: 2765      2894              dbg$gl_logfab [fab$v_nam] = 0;
: 2766      2895          END;
: 2767      2896
: 2768      2897      .message_vect = dbg$nmake_arg_vect
: 2769      2898          (shr$openout+dbg_fac_code, 3,
: 2770      2899          .msg_desc, .sts, .stv);
: 2771      2900
: 2772      2901      RETURN sts$k_severe;
: 2773      2902
: 2774      2903      END;
: 2775      2904
: 2776      2905      ! restore logging status and free temp storage
: 2777      2906
: 2778      2907      dbg$gb_def_out [out_log] = .log_temp;
: 2779      2908      IF .opened_log
: 2780      2909      THEN
: 2781      2910
: 2782      2911          ! Cleanup the "original" NAM block
: 2783      2912
: 2784      2913          BEGIN
: 2785      2914              dbg$rel_memory (.old_nam_ptr);
: 2786      2915              dbg$rel_memory (.dbg$gb_logfse);
: 2787      2916              dbg$rel_memory (.dbg$gb_logfsr);
: 2788      2917
: 2789      2918              dbg$gb_logfsr = .temp_fsr;
: 2790      2919              dbg$gb_logfse = .temp_fse;
: 2791      2920
: 2792      2921          ! Release the buffer that holds the log file spec. Since this
: 2793      2922          ! is a counted string we have to adjust the address to free
```



```

: 2794
: 2795
: 2796
: 2797
: 2798
: 2799
: 2800
: 2801
: 2802
2923
2924
2925
2926
2927
2928
2929
2930
2931

```

```

IF .fns NEQ 0
THEN
    dbg$rel_memory (.fna-1);
END;

RETURN sts$k_success;

END;

```

```

                                OFFC 00000
5E                                18 C2 00002
                                5A D4 00005
5B 00000000G 00 90 00007
                                00 94 0000E
                                00 B5 00014
                                03 12 0001A
                                00C7 31 0001C
5A                                01 D0 0001F 1$:
57 00000000G 00 D0 00022
6E 00000000G 00 90 00029
                                00 9F 00030
00000000G 00 01 FB 00036
59                                50 D0 0003D
45                                59 E8 00040
                                02 DD 00043
00000000G 00 01 FB 00045
                                00 D5 0004C
                                11 13 00052
04 60 00000000G 00 9B 00054
A0 00000000G 00 D0 0005B
                                0F 11 00063
04 60 00000000G 00 9B 00065 2$:
A0 00000000G 00 D0 0006C
7E 00000000G 00 7D 00074 3$:
                                50 DD 0007B
                                01 DD 0007D
                                00021058 8F DD 0007F
                                0319 31 00085
                                18 DD 00088 4$:
00000000G 00 01 FB 0008A
56                                50 D0 00091
0060 8F 00 00 2C 00094
                                66 0009B
66 6002 8F B0 0009C
7E 40 8F 9A 000A1
00000000G 00 01 FB 000A5
08 AE 50 D0 000AC
7E 40 8F 9A 000B0
00000000G 00 01 FB 000B4
04 AE 50 D0 000BB
0C A6 08 AE D0 000BF

```

```

.EXTRN SYSS$CLOSE, SYSS$PARSE
.ENTRY DBG$NSET_LOG, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 2695
R10,R11
SUBL2 #24, SP
CLRL OPENED_LOG 2740
MOVB DBG$GB_DEF_OUT, LOG_TEMP 2744
CLRB DBG$GB_DEF_OUT 2745
TSTW DBG$GL_LOGFAB+2 2750
BNEQ 1$
BRW 5$
MOVL #1, OPENED_LOG 2754
MOVL DBG$GL_LOGFAB+44, FNA 2755
MOVB DBG$GL_LOGFAB+52, FNS 2756
PUSHAB DBG$GL_LOGFAB 2758
CALLS #1, SYSS$CLOSE
MOVL R0, STATUS
BLBS STATUS, 4$ 2759
PUSHL #2 2766
CALLS #1, DBG$GET_TEMP_MEM
TSTL DBG$GL_LOG_BUF 2767
BEQL 2$
MOVZBW DBG$GL_LOGFAB+52, (MSG_DESC)
MOVL DBG$GL_LOGFAB+44, 4(MSG_DESC)
BRB 3$
MOVZBW DBG$GL_LOGFAB+53, (MSG_DESC)
MOVL DBG$GL_LOGFAB+48, 4(MSG_DESC)
MOVQ DBG$GL_LOGFAB+8, -(SP) 2771
PUSHL MSG_DESC 2770
PUSHL #1 2769
PUSHL #135256
BRW 34$
PUSHL #24 2780
CALLS #1, DBG$GET_MEMORY
MOVL R0, TEMP_NAM
MOVCS #0, (SP); #0, #96, (TEMP_NAM) 2781
MOVW #24578, (TEMP_NAM)
MOVZBL #64, -(SP) 2782
CALLS #1, DBG$GET_MEMORY
MOVL R0, TEMP_FSE
MOVZBL #64, -(SP) 2783
CALLS #1, DBG$GET_MEMORY
MOVL R0, TEMP_FSR
MOVL TEMP_FSE, 12(TEMP_NAM) 2784

```


04	A6	04	AE	D0	000C4	MOVL	TEMP FSR, 4(TEMP_NAM)	2785
0A	A6		01	8E	000C9	MNEGB	#1, TO(TEMP_NAM)	2786
02	A6		01	8E	000CD	MNEGB	#1, 2(TEMP_NAM)	2787
00000000G	00		56	D0	000D1	MOVL	TEMP_NAM, DBG\$GL_LOGFAB+40	2788
	58	00000000G	00	D0	000D8	MOVL	DBG\$GL_LOGNAM, OLD_NAM_PTR	2789
00000000G	00		56	D0	000DF	MOVL	TEMP_NAM, DBG\$GL_LOGNAM	2790
00000000G	00		02	8A	000E6	BICB2	#2, DBG\$GL_LOGFAB+7	2796
00000000G	00		02	88	000ED	BISB2	#2, DBG\$GL_LOGFAB+4	2797
00000000G	00		01	8A	000F4	BICB2	#1, DBG\$GL_LOGFAB+7	2798
	50	00000000G	00	D0	000FB	MOVL	DBG\$GL_LOG_BUF, R0	2802
			03	12	00102	BNEQ	6\$	
			016D	31	00104	BRW	21\$	
00000000G	00	01	A0	9E	00107	MOVAB	1(R0), DBG\$GL_LOGFAB+44	2809
00000000G	00		60	90	0010F	MOVB	(R0), DBG\$GL_LOGFAB+52	2810
		00000000G	00	9F	00116	PUSHAB	DBG\$GL_LOGFAB	2816
00000000G	00		01	FB	0011C	CALLS	#1, SYSSPARSE	
	59		50	D0	00123	MOVL	R0, STATUS	
	03		59	E9	00126	BLBC	STATUS, 7\$	2817
			0128	31	00129	BRW	19\$	
			02	DD	0012C	PUSHL	#2	2825
00000000G	00		01	FB	0012E	CALLS	#1, DBG\$GET_TEMPMEM	
	52		50	D0	00135	MOVL	R0, MSG_DESC	
	55	00000000G	00	D0	00138	MOVL	DBG\$GL_LOGFAB+8, STS	2829
	54	00000000G	00	D0	0013F	MOVL	DBG\$GL_LOGFAB+12, STV	2830
		00000000G	00	D5	00146	TSTL	DBG\$GL_LOG_BUF	2831
			11	13	0014C	BEQL	8\$	
	62	00000000G	00	9B	0014E	MOVZBW	DBG\$GL_LOGFAB+52, (MSG_DESC)	
04	A2	00000000G	00	D0	00155	MOVL	DBG\$GL_LOGFAB+44, 4(MSG_DESC)	
			0F	11	0015D	BRB	9\$	
	62	00000000G	00	9B	0015F	MOVZBW	DBG\$GL_LOGFAB+53, (MSG_DESC)	
04	A2	00000000G	00	D0	00166	MOVL	DBG\$GL_LOGFAB+48, 4(MSG_DESC)	
	03		5A	E8	0016E	BLBS	OPENED_LOG, 10\$	2833
			00BF	31	00171	BRW	17\$	
			56	DD	00174	PUSHL	TEMP_NAM	2834
00000000G	00		01	FB	00176	CALLS	#1, DBG\$REL_MEMORY	
		04	AE	DD	0017D	PUSHL	TEMP_FSR	
00000000G	00		01	FB	00180	CALLS	#1, DBG\$REL_MEMORY	
		08	AE	DD	00187	PUSHL	TEMP_FSE	
00000000G	00		01	FB	0018A	CALLS	#1, DBG\$REL_MEMORY	
00000000G	00		58	D0	00191	MOVL	OLD_NAM_PTR, DBG\$GL_LOGNAM	
			58	D4	00198	CLRL	OLD_NAM_PTR	
00000000G	00	00000000G	00	D0	0019A	MOVL	DBG\$GL_LOGNAM, DBG\$GL_LOGFAB+40	
00000000G	00		57	D0	001A5	MOVL	FNA, DBG\$GL_LOGFAB+44	
00000000G	00		6E	90	001AC	MOVB	FNS, DBG\$GL_LOGFAB+52	
00000000G	00		03	88	001B3	BISB2	#3, DBG\$GL_LOGFAB+7	
00000000G	00		02	8A	001BA	BICB2	#2, DBG\$GL_LOGFAB+4	
		0C	AE	9F	001C1	PUSHAB	TYPE_B	
0000V	CF		01	FB	001C4	CALLS	#1, DBG\$NSETUP_LOG	
	5E		50	E8	001C9	BLBS	ERR, 16\$	
			02	DD	001CC	PUSHL	#2	
00000000G	00		01	FB	001CE	CALLS	#1, DBG\$GET_TEMPMEM	
		00000000G	00	D5	001D5	TSTL	DBG\$GL_LOG_BUF	
			11	13	001DB	BEQL	11\$	
	60	00000000G	00	9B	001DD	MOVZBW	DBG\$GL_LOGFAB+52, (MSG_DESC)	
04	A0	00000000G	00	D0	001E4	MOVL	DBG\$GL_LOGFAB+44, 4(MSG_DESC)	
			0F	11	001EC	BRB	12\$	
	60	00000000G	00	9B	001EE	MOVZBW	DBG\$GL_LOGFAB+53, (MSG_DESC)	11\$:

04	A0	00000000G	00	D0	001F5	MOVL	DBG\$GL_LOGFAB+48, 4(MSG_DESC)		
	01	0C	AE	D1	001FD	12\$:	CMPL	TYPE_B, #1	
			10	12	00201	13\$:	BNEQ	14\$	
	53	00000000G	00	D0	00203		MOVL	DBG\$GL_LOGFAB+8, STS	
	51	00000000G	00	D0	0020A		MOVL	DBG\$GL_LOGFAB+12, STV	
			0E	11	00211		BRB	15\$	
	53	00000000G	00	D0	00213	14\$:	MOVL	DBG\$GL_LOGRAB+8, STS	
	51	00000000G	00	D0	0021A		MOVL	DBG\$GL_LOGRAB+12, STV	
			51	DD	00221	15\$:	PUSHL	STV	
			09	BB	00223		PUSHR	#*M<R0,R3>	
			01	DD	00225		PUSHL	#1	
			0171	31	00227		BRW	33\$	
00000000G	00		5B	90	0022A	16\$:	MOVW	LOG_TEMP, DBG\$GB_DEF_OUT	
			12	11	00231		BRB	18\$	2833
		00000000G	00	D4	00233	17\$:	CLRL	DBG\$GL_LOGFAB+44	2838
		00000000G	00	94	00239		CLRB	DBG\$GL_LOGFAB+52	2839
		00000000G	00	D4	0023F		CLRL	DBG\$GL_LOG_BUF	2840
			54	DD	00245	18\$:	PUSHL	STV	2845
			24	BB	00247		PUSHR	#*M<R2,R5>	
			01	DD	00249		PUSHL	#1	2844
		000210F8	8F	DD	0024B		PUSHL	#135416	
			014D	31	00251		BRW	34\$	
	50	00000000G	00	D0	00254	19\$:	MOVL	DBG\$GL_LOGNAM, R0	2851
	0E	34	A0	E9	0025B		BLBC	52(R0), 20\$	
00000000G	00		02	88	0025F		BISB2	#2, DBG\$GL_LOGFAB+7	2854
00000000G	00		02	8A	00266		BICB2	#2, DBG\$GL_LOGFAB+4	2855
00000000G	00		01	88	0026D	20\$:	BISB2	#1, DBG\$GL_LOGFAB+7	2858
		10	AE	9F	00274	21\$:	PUSHAB	B TYPE	2863
0000V	CF		01	FB	00277		CALLS	#T, DBG\$NSETUP_LOG	
	59		50	D0	0027C		MOVL	R0, STATUS	
	03		59	E9	0027F		BLBC	STATUS, 22\$	2864
			012B	31	00282		BRW	35\$	
			02	DD	00285	22\$:	PUSHL	#2	2872
00000000G	00		01	FB	00287		CALLS	#1, DBG\$GET_TEMP MEM	
	55		50	D0	0028E		MOVL	R0, MSG_DESC	
		00000000G	00	D5	00291		TSTL	DBG\$GL_LOG_BUF	2873
			11	13	00297		BEQL	23\$	
	65	00000000G	00	9B	00299		MOVZBW	DBG\$GL_LOGFAB+52, (MSG_DESC)	
04	A5	00000000G	00	D0	002A0		MOVL	DBG\$GL_LOGFAB+44, 4(MSG_DESC)	
			0F	11	002A8		BRB	24\$	
	65	00000000G	00	9B	002AA	23\$:	MOVZBW	DBG\$GL_LOGFAB+53, (MSG_DESC)	
04	A5	00000000G	00	D0	002B1		MOVL	DBG\$GL_LOGFAB+48, 4(MSG_DESC)	
	01	10	AE	D1	002B9	24\$:	CMPL	B TYPE, #1	2874
			10	12	002BD		BNEQ	25\$	
	52	00000000G	00	D0	002BF		MOVL	DBG\$GL_LOGFAB+8, STS	2877
	54	00000000G	00	D0	002C6		MOVL	DBG\$GL_LOGFAB+12, STV	2878
			0E	11	002CD		BRB	26\$	2874
	52	00000000G	00	D0	002CF	25\$:	MOVL	DBG\$GL_LOGRAB+8, STS	2882
	54	00000000G	00	D0	002D6		MOVL	DBG\$GL_LOGRAB+12, STV	2883
	03		5A	E8	002DD	26\$:	BLBS	OPENED_LOG, 27\$	2886
			0099	31	002E0		BRW	31\$	
			56	DD	002E3	27\$:	PUSHL	TEMP NAM	2887
00000000G	00		01	FB	002E5		CALLS	#1, DBG\$REL_MEMORY	
		04	AE	DD	002EC		PUSHL	TEMP_FSR	
00000000G	00		01	FB	002EF		CALLS	#1, DBG\$REL_MEMORY	
		08	AE	DD	002F6		PUSHL	TEMP_FSE	
00000000G	00		01	FB	002F9		CALLS	#1, DBG\$REL_MEMORY	

00000000G	00		58	D0	00300	MOVL	OLD_NAM_PTR, DBG\$GL_LOGNAM	
			58	D4	00307	CLRL	OLD_NAM_PTR	
00000000G	00	00000000G	00	D0	00309	MOVL	DBG\$GL_LOGNAM, DBG\$GL_LOGFAB+40	
00000000G	00		57	D0	00314	MOVL	FNA, DBG\$GL_LOGFAB+44	
00000000G	00		6E	90	0031B	MOVB	FNS, DBG\$GL_LOGFAB+52	
00000000G	00		03	88	00322	BISB2	#3, DBG\$GL_LOGFAB+7	
00000000G	00		02	8A	00329	BICB2	#2, DBG\$GL_LOGFAB+4	
		14	AE	9F	00330	PUSHAB	TYPE_B	
0000V	CF		01	FB	00333	CALLS	#1, DBG\$NSETUP_LOG	
	38		50	E8	00338	BLBS	ERR, 30\$	
			02	DD	0033B	PUSHL	#2	
00000000G	00		01	FB	0033D	CALLS	#1, DBG\$GET_TEMP_MEM	
		00000000G	00	D5	00344	TSTL	DBG\$GL_LOG_BUF	
			11	13	0034A	BEQL	28\$	
	60	00000000G	00	9B	0034C	MOVZBW	DBG\$GL_LOGFAB+52, (MSG_DESC)	
04	A0	00000000G	00	D0	00353	MOVL	DBG\$GL_LOGFAB+44, 4(MSG_DESC)	
			0F	11	0035B	BRB	29\$	
	60	00000000G	00	9B	0035D	MOVZBW	DBG\$GL_LOGFAB+53, (MSG_DESC)	
04	A0	00000000G	00	D0	00364	MOVL	DBG\$GL_LOGFAB+48, 4(MSG_DESC)	
	01	14	AE	D1	0036C	CMPL	TYPE_B, #1	
			FE	8E	31	BRW	13\$	
00000000G	00		5B	90	00373	MOVB	LOG_TEMP, DBG\$GB_DEF_OUT	
			19	11	0037A	BRB	32\$	2886
		00000000G	00	D4	0037C	CLRL	DBG\$GL_LOGFAB+44	2891
		00000000G	00	94	00382	CLRB	DBG\$GL_LOGFAB+52	2892
		00000000G	00	D4	00388	CLRL	DBG\$GL_LOG_BUF	2893
00000000G	00		01	8A	0038E	BICB2	#1, DBG\$GL_LOGFAB+7	2894
			14	BB	00395	PUSHR	#M<R2,R4>	2899
			55	DD	00397	PUSHL	MSG_DESC	
			03	DD	00399	PUSHL	#3	2898
		000210A0	8F	DD	0039B	PUSHL	#135328	
00000000G	00		05	FB	003A1	CALLS	#5, DBG\$NMAKE_ARG_VECT	
04	BC		50	D0	003A8	MOVL	R0, @MESSAGE_VECT	
	50		04	D0	003AC	MOVL	#4, R0	2901
				04	003AF	RET		
00000000G	00		5B	90	003B0	MOVB	LOG_TEMP, DBG\$GB_DEF_OUT	2907
	41		5A	E9	003B7	BLBC	OPENED_LOG, 36\$	2908
			58	DD	003BA	PUSHL	OLD_NAM_PTR	2914
00000000G	00		01	FB	003BC	CALLS	#1, DBG\$REL_MEMORY	
		00000000G	00	DD	003C3	PUSHL	DBG\$GB_LOGFSE	2915
00000000G	00		01	FB	003C9	CALLS	#1, DBG\$REL_MEMORY	
		00000000G	00	DD	003D0	PUSHL	DBG\$GB_LOGFSR	2916
00000000G	00		01	FB	003D6	CALLS	#1, DBG\$REL_MEMORY	
00000000G	00	04	AE	D0	003DD	MOVL	TEMP_FSR, DBG\$GB_LOGFSR	2918
00000000G	00	08	AE	D0	003E5	MOVL	TEMP_FSE, DBG\$GB_LOGFSE	2919
			6E	95	003ED	TSTB	FNS	2924
			0A	13	003EF	BEQL	36\$	
		FF	A7	9F	003F1	PUSHAB	-1(FNA)	2926
00000000G	00		01	FB	003F4	CALLS	#1, DBG\$REL_MEMORY	
	50		01	D0	003FB	MOVL	#1, R0	2929
			04	003FE	RET			2931

; Routine Size: 1023 bytes, Routine Base: DBG\$CODE + 15B9


```
2804 2932 1 ROUTINE DBG$NSETUP_LOG (BLOCK_TYPE) =
2805 2933 1 ++
2806 2934 1 FUNCTIONAL DESCRIPTION:
2807 2935 1 This routine creates and opens the log file.
2808 2936 1
2809 2937 1 FORMAL PARAMETERS:
2810 2938 1
2811 2939 1 block_type - indicates whether error to report is for
2812 2940 1 FAB ( = 1) or RAB ( = 2)
2813 2941 1
2814 2942 1 IMPLICIT INPUTS:
2815 2943 1
2816 2944 1 dbg$gl_log_buf - pointer to log filespec buffer
2817 2945 1
2818 2946 1 IMPLICIT OUTPUTS:
2819 2947 1 The resultant LOG file-spec as resolved by RMS
2820 2948 1
2821 2949 1 ROUTINE VALUE:
2822 2950 1 Returns the RMS status code if there's an error, otherwise returns TRUE
2823 2951 1
2824 2952 1 SIDE EFFECTS:
2825 2953 1 The log file is created and a RAB connected to it.
2826 2954 1 --
2827 2955 1
2828 2956 2 BEGIN
2829 2957 2
2830 2958 2 LOCAL
2831 2959 2 STATUS;
2832 2960 2 MAP
2833 2961 2 DBG$GL_LOG_BUF : REF VECTOR [,BYTE];
2834 2962 2
2835 2963 2 ! See if a log file is already open
2836 2964 2
2837 2965 2 IF .dbg$gl_logfab [fab$w_ifi] NEQ 0
2838 2966 2 THEN
2839 2967 2 RETURN sts$k_success;
2840 2968 2
2841 2969 2 .block_type = 1;
2842 2970 2 status = $CREATE(FAB = dbg$gl_logfab);
2843 2971 2
2844 2972 2 IF NOT .status
2845 2973 2 THEN
2846 2974 2 RETURN (.status) ;
2847 2975 2
2848 2976 2 dbg$gl_lograb [rab$l_fab] = dbg$gl_logfab;
2849 2977 2 .block_type = 2;
2850 2978 2 status = $CONNECT (RAB = dbg$gl_lograb);
2851 2979 2
2852 2980 2 IF NOT .status
2853 2981 2 THEN
2854 2982 2 RETURN (.status) ;
2855 2983 2
2856 2984 2
2857 2985 2 ! Eventually put out the title line here
2858 2986 2
2859 2987 2 RETURN true;
2860 2988 2
```


DBGNSET
V04-000

E 5
16-Sep-1984 01:58:19
14-Sep-1984 12:17:21

VAX-11 Bliss-32 V4.0-742
[DEBUG.SRC]DBGNSET.B32;1

Page 91
(6)

: 2861 2989 1 END; ! END OF DBG\$NSET_UP_LOG

.EXTRN SYSS\$CREATE, SYSS\$CONNECT

0004 00000 DBG\$NSETUP LOG:						
	52	00000000G	00 9E 00002	WORD	Save R2	: 2932
		02	A2 B5 00009	MOVAB	DBG\$GL_LOGFAB, R2	
			2B 12 0000C	TSTW	DBG\$GL_LOGFAB+2	: 2965
04	BC		01 D0 0000E	BNEQ	1\$	: 2969
			52 DD 00012	MOVL	#1, @BLOCK_TYPE	: 2970
00000000G	00		01 FB 00014	PUSHL	R2	
	1E		50 E9 0001B	CALLS	#1, SYSS\$CREATE	: 2972
00000000G	00		62 9E 0001E	BLBC	STATUS, 2\$	: 2976
04	BC		02 D0 00025	MOVAB	DBG\$GL_LOGFAB, DBG\$GL_LOGRAB+60	: 2977
		00000000G	00 9F 00029	MOVL	#2, @BLOCK_TYPE	: 2978
00000000G	00		01 FB 0002F	PUSHAB	DBG\$GL_LOGRAB	
	03		50 E9 00036	CALLS	#1, SYSS\$CONNECT	: 2980
	50		01 D0 00039 1\$:	BLBC	STATUS, 2\$	: 2987
			04 0003C 2\$:	MOVL	#1, R0	: 2989
				RET		

: Routine Size: 61 bytes, Routine Base: DBG\$CODE + 19B8


```

: 2863      2990 1 ROUTINE DUMMY(X): NOVALUE =
: 2864      2991 1
: 2865      2992 1 FUNCTION
: 2866      2993 1 This is a do-nothing routine that is called from within DBG$NPARSE_SET
: 2867      2994 1 in order to foil an incorrect BLISS compiler optimization.
: 2868      2995 1
: 2869      2996 2 BEGIN
: 2870      2997 2 0
: 2871      2998 1 END;

```

```

0000 00000 DUMMY: .WORD Save nothing
04 00002 RET

```

```

: 2990
: 2998

```

; Routine Size: 3 bytes, Routine Base: DBG\$CODE + 19F5

```

: 2872      2999 1 END ! End of module
: 2873      3000 0 ELUDOM

```

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
DBG\$GLOBAL	4	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
DBG\$PLIT	865	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(0)
DBG\$OWN	12	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
DBG\$CODE	6648	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(0)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	62	0	1000	00:01.9
-\$255\$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1	32	0	0	7	00:00.1
-\$255\$DUA28:[DEBUG.OBJ]DBGLIB.L32;1	1545	48	3	97	00:02.0
-\$255\$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1	418	13	3	31	00:00.3
-\$255\$DUA28:[DEBUG.OBJ]DBGMSG.L32;1	386	17	4	22	00:00.3
-\$255\$DUA28:[DEBUG.OBJ]DBGGEN.L32;1	150	13	8	12	00:00.3

DBGNSET
V04-000

G 5
16-Sep-1984 01:58:19
14-Sep-1984 12:17:21

VAX-11 Bliss-32 V4.0-742
[DEBUG.SRC]DBGNSET.B32;1

Page 93
(7)

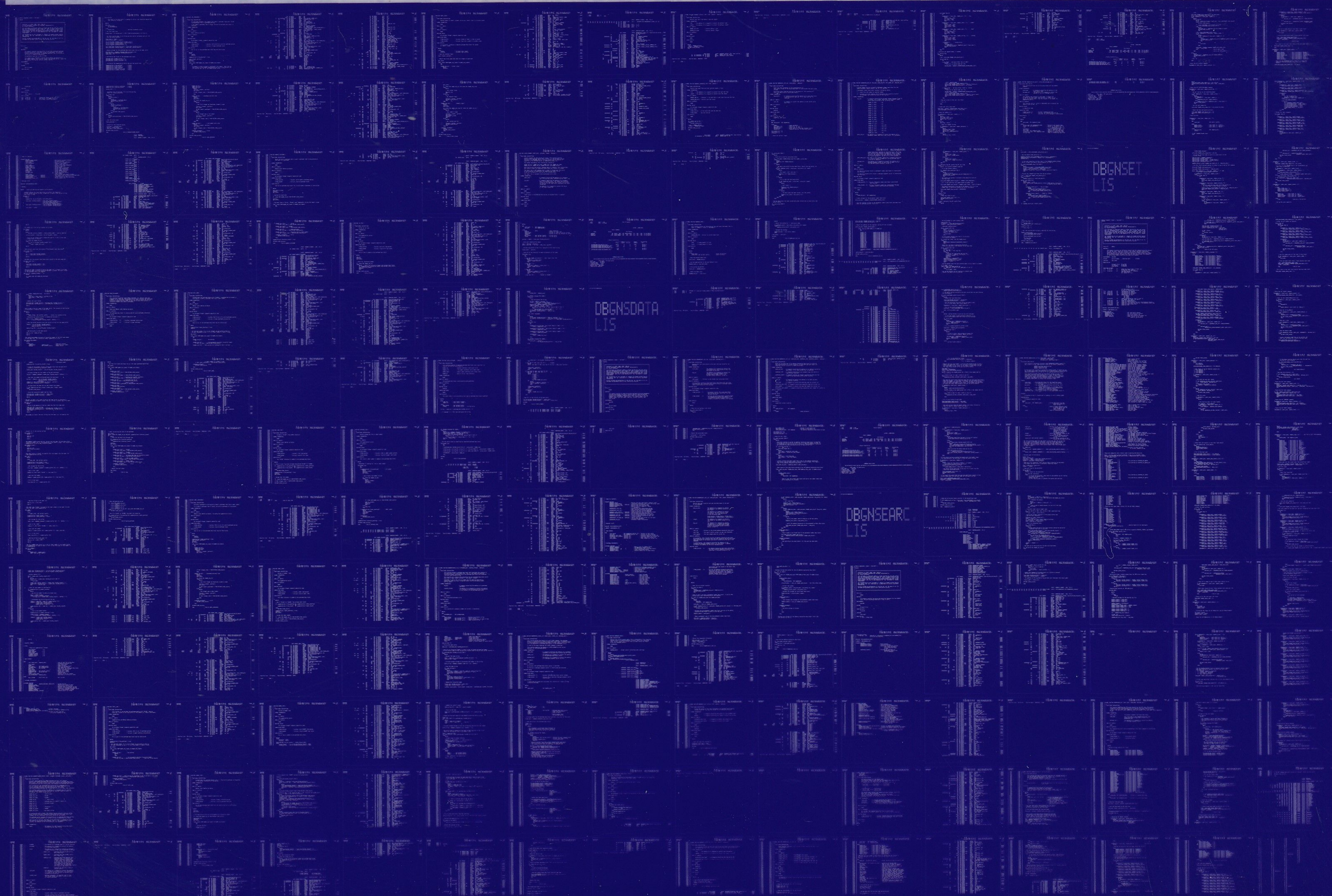
COMMAND QUALIFIERS

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

: Size: 6648 code + 881 data bytes
: Run Time: 01:54.5
: Elapsed Time: 05:45.4
: Lines/CPU Min: 1571
: Lexemes/CPU-Min: 12554
: Memory Used: 1089 pages
: Compilation Complete

0088 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0089

AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY