

DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBBBB	UUU	UUU	GGGGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBBBB	UUU	UUU	GGGGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBBBB	UUU	UUU	GGGGGGGGGGGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	EEEEEEEEEEEEEE	BBBBBBBBBBBBBB	UUU	UUU	GGG
DDD	EEEEEEEEEEEEEE	BBBBBBBBBBBBBB	UUU	UUU	GGG
DDD	EEEEEEEEEEEEEE	BBBBBBBBBBBBBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBBBB	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBBBB	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBBBB	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	GGGGGGGGGG

```

DDDDDDDD  BBBB8888  GGGGGGGG  NN      NN  TTTTTTTTTT  YY      YY  PPPPPPPP  EEEEEEEEEE
DDDDDDDD  BBBB8888  GGGGGGGG  NN      NN  TTTTTTTTTT  YY      YY  PPPPPPPP  EEEEEEEEEE
DD      DD  BB      BB  GG      GG  NN      NN  TT      TT  YY      YY  PP      PP  EE
DD      DD  BB      BB  GG      GG  NN      NN  TT      TT  YY      YY  PP      PP  EE
DD      DD  BB      BB  GG      GG  NNNN     NN  TT      TT  YY      YY  PP      PP  EE
DD      DD  BB      BB  GG      GG  NNNN     NN  TT      TT  YY      YY  PP      PP  EE
DD      DD  BBBB8888  GG      GG  NN      NN  TT      TT  YY      YY  PPPPPPPP  EEEEEEEEEE
DD      DD  BBBB8888  GG      GG  NN      NN  TT      TT  YY      YY  PPPPPPPP  EEEEEEEEEE
DD      DD  BB      BB  GG  GGGGGG  NN      NNNN  TT      TT  YY      YY  PP      PP  EE
DD      DD  BB      BB  GG  GGGGGG  NN      NNNN  TT      TT  YY      YY  PP      PP  EE
DD      DD  BB      BB  GG      GG  NN      NN  TT      TT  YY      YY  PP      PP  EE
DD      DD  BB      BB  GG      GG  NN      NN  TT      TT  YY      YY  PP      PP  EE
DDDDDDDD  BBBB8888  GGGGGG  NN      NN  TT      TT  YY      YY  PP      PP  EEEEEEEEEE
DDDDDDDD  BBBB8888  GGGGGG  NN      NN  TT      TT  YY      YY  PP      PP  EEEEEEEEEE

```

```

LL      LL  IIIIII  SSSSSSSS
LL      LL  IIIIII  SSSSSSSS
LL      LL  II      SS
LL      LL  II      SS
LL      LL  II      SS
LL      LL  II      SS
LL      LL  II      SSSSSS
LL      LL  II      SSSSSS
LL      LL  II      SS
LL      LL  II      SS
LL      LL  II      SS
LL      LL  II      SS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

```
0001 0 MODULE DBGNTYPE (IDENT = 'V04-000') =
0002 0
0003 1 BEGIN
0004 1
0005 1
0006 1 *****
0007 1 *
0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
0010 1 * ALL RIGHTS RESERVED. *
0011 1 *
0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
0017 1 * TRANSFERRED. *
0018 1 *
0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
0021 1 * CORPORATION. *
0022 1 *
0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
0025 1 *
0026 1 *****
0027 1 *****
0028 1
0029 1
0030 1 ++
0031 1 FACILITY:
0032 1
0033 1     DEBUG
0034 1
0035 1 ABSTRACT:
0036 1
0037 1     This module contains the parse and execution networks for the TYPE
0038 1     verb. The parsing method employed is that of ATN's.
0039 1
0040 1 ENVIRONMENT:
0041 1
0042 1     VAX/VMS
0043 1
0044 1 AUTHOR:
0045 1
0046 1     Richard Title
0047 1
0048 1 CREATION DATE:
0049 1
0050 1     9-14-81
0051 1
0052 1 VERSION:
0053 1
0054 1     V03.0-001
0055 1
0056 1 MODIFIED BY:
0057 1     V. Holt           May, 1982
```

```
.. 58      0058 1 |  
.. 59      0059 1 | REVISION HISTORY:  
.. 60      0060 1 | 3.10 14-May-82  
.. 61      0061 1 |  
.. 62      0062 1 | 3.11 2-Jun-82  
.. 63      0063 1 |  
.. 64      0064 1 |  
.. 65      0065 1 | --
```

VJH

Added call to DBG\$FLUSHBUF, eliminating need to initialize local output buffer.

VJH

Removed all references to DBG\$FAO_PUT and DBG\$OUT_PUT, replacing them with calls to DBG\$PRINT and DBG\$NEWLINE, respectively.

67	0066	1	!		
68	0067	1	!	TABLE OF CONTENTS:	
69	0068	1	!		
70	0069	1	!		
71	0070	1	!	FORWARD ROUTINE	
72	0071	1	!	DBG\$NPARSE_TYPE,	Parse routine for TYPE
73	0072	1	!	DBG\$NEXECUTE_TYPE,	Execution routine for TYPE
74	0073	1	!	DBG\$NPARSE_EDIT,	Parse routine for EDIT
75	0074	1	!	DBG\$NEXECUTE_EDIT;	Execution routine for EDIT
76	0075	1	!		
77	0076	1	!	!	
78	0077	1	!	REQUIRE FILES:	
79	0078	1	!		
80	0079	1	!		
81	0080	1	!	REQUIRE 'SRC\$:DBGPROLOG.REQ';	
82	0214	1	!		
83	0215	1	!	LIBRARY 'LIB\$:DBGGEN.L32';	
84	0216	1	!		
85	0217	1	!	EXTERNAL ROUTINE	
86	0218	1	!	dbg\$sta_getsourcmod,	Gets module rst pointer
87	0219	1	!	dbg\$sta_symname,	Turns module rst pointer back into a string
88	0220	1	!		
89	0221	1	!	dbg\$print: NOVALUE,	Formats ascii output into output buf
90	0222	1	!	dbg\$flushbuf: NOVALUE,	Initializes new print line
91	0223	1	!	dbg\$next_word,	gets next word from input
92	0224	1	!	dbg\$newline: NOVALUE,	outputs the output buffer
93	0225	1	!	dbg\$nmatch,	Tries to match the next token
94	0226	1	!	dbg\$get_tempmem,	Allocates space
95	0227	1	!	dbg\$make_arg_vect,	Constructs error messages
96	0228	1	!	dbg\$nsave_decimal_integer,	Reads an integer from the input string
97	0229	1	!		
98	0230	1	!	dbg\$nsyntax_error,	Reports a syntax error
99	0231	1	!		
100	0232	1	!	dbg\$scr_screen_normal: NOVALUE,	Set screen to normal in screen mode
101	0233	1	!	dbg\$src_line_to_rec,	Translate line number to record num
102	0234	1	!	dbg\$src_type_lnum_source,	The routine in DBGSOURCE that actually outputs the source lines to the terminal
103	0235	1	!		
104	0236	1	!		
105	0237	1	!	lib\$do_command,	Execute DCL command
106	0238	1	!	lib\$spawn,	Execute DCL command
107	0239	1	!	smg\$set_keypad_mode,	Set terminal to numeric/application mode
108	0240	1	!	sys\$fao:	Formatted ASCII output
109	0241	1	!		
110	0242	1	!	EXTERNAL	
111	0243	1	!	DBG\$GB_KEYPAD_INPUT: BYTE,	Set if mode is KEYPAD
112	0244	1	!	DBG\$GL_KEYBOARD_ID,	Used by SMG\$SET_KEYPAD_MODE
113	0245	1	!	DBG\$GV_CONTROL:~DBG\$CONTROL_FLAGS,	Global status flags
114	0246	1	!	DBG\$GL_DEVELOPER: BITVECTOR[],	Developer bits
115	0247	1	!	DBG\$GL_SCREEN_SOURCE,	Non-zero pointer if source is being directed to a screen display
116	0248	1	!		
117	0249	1	!	DBG\$SRC_NEXT_MODRSTPTR,	Contains module RST pointer used by TYPE command
118	0250	1	!		
119	0251	1	!	DBG\$SRC_NEXT_LNUM;	Contains next line number to be output in case of a TYPE command with no argument
120	0252	1	!		
121	0253	1	!		
122	0254	1	!		
123	0255	1	!	! These values can go in the VERB_COMPOSITE field of the EDIT verb	

DBGNTYPE
V04-000

F 11
16-Sep-1984 02:08:23
14-Sep-1984 12:17:25

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

Page 4
(2)

DBI
V04

```
: 124      0256 1 !  
: 125      0257 1 LITERAL  
: 126      0258 1     EDIT_EXIT = 1;    ! EDIT/EXIT  
: 127      0259 1     EDIT_NOEXIT = 2; ! EDIT/NOEXIT  
: 128      0260 1
```

.....

```

130 0261 1 GLOBAL ROUTINE DBG$NPARSE_TYPE(INPUT_DESC, VERB_NODE, MESSAGE_VECT) =
131 0262 1
132 0263 1 Functional Description
133 0264 1
134 0265 1     ATN parse network for the TYPE verb.
135 0266 1     This routine takes a verb node for the TYPE verb, and a string
136 0267 1     descriptor for the remaining (unparsed) input.
137 0268 1     It constructs a linked list of noun nodes, each of which
138 0269 1     contains a module rst pointer and a line number range.
139 0270 1
140 0271 1 Formal Parameters
141 0272 1
142 0273 1     INPUT_DESC     - A longword containing the address of the
143 0274 1                   command input descriptor.
144 0275 1
145 0276 1     VERB_NODE      - A longword containing the address of the verb node
146 0277 1
147 0278 1     MESSAGE_VECT   - The address of a longword to contain the address
148 0279 1                   of a standard message argument vector.
149 0280 1
150 0281 1 Implicit Inputs
151 0282 1
152 0283 1     None
153 0284 1
154 0285 1 Implicit Outputs
155 0286 1
156 0287 1     On success, the command execution tree is constructed.
157 0288 1     On failure, a message argument vector is constructed or obtained.
158 0289 1
159 0290 1 Routine value
160 0291 1
161 0292 1     ST$K_SUCCESS (1)  - Success. Command execution tree constructed.
162 0293 1     ST$K_SEVERE  (4)  - Failure. Error encountered. Message argument
163 0294 1                       constructed and returned.
164 0295 1
165 0296 1 Side Effects
166 0297 1
167 0298 1     None
168 0299 1
169 0300 1
170 0301 2 BEGIN
171 0302 2
172 0303 2 MAP
173 0304 2     input_desc  : REF dbg$stg_desc,
174 0305 2     verb_node   : REF dbg$verb_node;
175 0306 2
176 0307 2 BIND
177 0308 2     dbg$cs_colon      = UPLIT BYTE (1, dbg$k_colon),
178 0309 2     dbg$cs_slash     = UPLIT BYTE (1, dbg$k_slash),
179 0310 2     dbg$cs_comma     = UPLIT BYTE (1, dbg$k_comma),
180 0311 2     dbg$cs_cr        = UPLIT BYTE (1, dbg$k_car_return),
181 0312 2     dbg$cs_backslash = UPLIT BYTE (1, dbg$k_backslash);
182 0313 2
183 0314 2 LOCAL
184 0315 2     modrstptr,
185 0316 2     noun_node      : REF dbg$noun_node,
186 0317 2     prev_noun_node : REF dbg$noun_node,

```

```

187 0318 2 low lnum,
188 0319 2 high lnum,
189 0320 2 first_flag;
190 0321 2 ! True if this is the first
191 0322 2 ! line number range in the list.
192 0323 2
193 0324 2 ! Create and link a noun node
194 0325 2 noun_node = dbg$get_tempmem(dbg$k_noun_node_size);
195 0326 2 verb_node[dbg$l_verb_object_ptr] = .noun_node;
196 0327 2
197 0328 2 first_flag = TRUE;
198 0329 2 modrstptr = 0;
199 0330 2
200 0331 2 ! Check for TYPE <cr>
201 0332 2
202 0333 2 IF dbg$match(.input_desc, dbg$cs_cr, 1)
203 0334 2 THEN
204 0335 2
205 0336 2 BEGIN ! no argument supplied
206 0337 2
207 0338 2 ! Try to get default module.
208 0339 2
209 0340 2 IF .dbg$src_next_modrstptr EQL 0
210 0341 2 THEN
211 0342 2 BEGIN
212 0343 2 ! report an error
213 0344 2 .message_vect = dbg$make_arg_vect(dbg$_nonxtlin);
214 0345 2 RETURN sfs$k_severe;
215 0346 2 END;
216 0347 2
217 0348 2 ! The module rst pointer is placed in the adjective field of
218 0349 2 ! the noun node.
219 0350 2
220 0351 2 noun_node [dbg$l_adjective_ptr] = .dbg$src_next_modrstptr;
221 0352 2
222 0353 2 ! Fill in the line numbers based on global info
223 0354 2
224 0355 2 noun_node [dbg$l_noun_value] = .dbg$src_next_lnum;
225 0356 2 noun_node [dbg$l_noun_value2] = .dbg$src_next_h_lnum;
226 0357 2
227 0358 2 ! The link field is zero
228 0359 2
229 0360 2 noun_node [dbg$l_noun_link] = 0;
230 0361 2
231 0362 2 END ! no argument supplied
232 0363 2
233 0364 2 ELSE
234 0365 2
235 0366 2 WHILE TRUE DO
236 0367 2
237 0368 2 BEGIN
238 0369 2
239 0370 2 OWN
240 0371 2 name_buf : VECTOR [81, BYTE]; ! Holds counted string with
241 0372 2 ! the module name
242 0373 2
243 0374 2

```

```

244 0375 LOCAL
245 0376     string_ptr,
246 0377     length,
247 0378     char;
248 0379
249 0380 ! Loop through the list of line number ranges.
250 0381 ! The most general form of the command is:
251 0382
252 0383 ! TYPE mod1\l1:h1, mod2\l2:h2, ... , modn\ln:hn
253 0384
254 0385 ! The user may omit the module, in which case it
255 0386 ! defaults to a previously-specified module or
256 0387 ! to a module determined by the current scope.
257 0388 ! He may omit the high line num, in which case it
258 0389 ! is the same as the low line num.
259 0390
260 0391 name_buf[0] = 0;
261 0392 string_ptr = .input_desc[dsc$a_pointer];
262 0393 length = .input_desc[dsc$w_length];
263 0394
264 0395 ! read past leading blanks
265 0396
266 0397 WHILE .length GTR 0
267 0398 DO
268 0399     BEGIN
269 0400     char = ch$rchar a(string_ptr);
270 0401     length = .length - 1;
271 0402     IF .char NEQ dbg$k_blank
272 0403     THEN
273 0404         EXITLOOP;
274 0405     END;
275 0406
276 0407 ! If the length reaches zero then it is an error
277 0408
278 0409 IF .length EQL 0 AND .char EQL dbg$k_blank
279 0410 THEN
280 0411     BEGIN
281 0412     .message_vect = dbg$make_arg_vect(dbg$_needmore);
282 0413     RETURN sts$k_severe;
283 0414     END;
284 0415
285 0416 ! Read until we reach a separating character.
286 0417 ! Place the characters into name_buf as we read them.
287 0418
288 0419 WHILE .length GTR 0
289 0420 DO
290 0421     BEGIN
291 0422     IF .char EQL '\\' OR .char EQL ','
292 0423     OR .char EQL ':' OR .char EQL ';'
293 0424     THEN
294 0425         EXITLOOP;
295 0426     name_buf[0] = .name_buf[0] + 1;
296 0427     name_buf[name_buf[0]] = .char;
297 0428     char = ch$rchar a(string_ptr);
298 0429     length = .length - 1;
299 0430     END;
300 0431

```

```

301      0432 3      IF .char EQL '\ ' ! this signifies that we read a module name
302      0433 3      THEN
303      0434 4          BEGIN
304      0435 4              ! Update the input descriptor
305      0436 4              input_desc[dsc$a_pointer] = .string_ptr;
306      0437 4              input_desc[dsc$w_length] = .length;
307      0438 4
308      0439 4              ! convert the name to an rst pointer and put it in the
309      0440 4              ! adjective field.
310      0441 4
311      0442 4              noun_node[dbg$l_adjective_ptr] =
312      0443 4                  dbg$sta_getsourcmod(name_buf);
313      0444 4
314      0445 4              ! If the above routine returned zero then the user entered
315      0446 4              ! an invalid module.
316      0447 4
317      0448 4
318      0449 4              IF .noun_node[dbg$l_adjective_ptr] EQ 0
319      0450 4              THEN
320      0451 5                  BEGIN
321      0452 5                      .message_vect = dbg$nmake_arg_vect(
322      0453 5                          dbg$nosuchmodu, 1, name_buf);
323      0454 5                      RETURN stf$sk_severe;
324      0455 5                  END;
325      0456 4
326      0457 4              ! fill in new value of modrstptr
327      0458 4              modrstptr = .noun_node[dbg$l_adjective_ptr];
328      0459 4
329      0460 4              END ! pick up module name
330      0461 4
331      0462 3      ELSE
332      0463 3          BEGIN ! fill in default module
333      0464 4
334      0465 4              IF .modrstptr NEQ 0
335      0466 4              THEN
336      0467 4                  ! use global default
337      0468 4                  noun_node[dbg$l_adjective_ptr] = .modrstptr
338      0469 4
339      0470 4              ELSE
340      0471 4
341      0472 4                  BEGIN
342      0473 5                      ! No default in modrstptr, so try to fill
343      0474 5                      ! in a module based on current scope
344      0475 5
345      0476 5                      modrstptr = dbg$sta_getsourcmod(0);
346      0477 5
347      0478 5                      IF .modrstptr EQ 0
348      0479 5                      THEN
349      0480 5                          BEGIN
350      0481 5                              BEGIN
351      0482 6                                  ! If this is still zero, then we have no scope with
352      0483 6                                  ! which to supply a module. Report an error.
353      0484 6
354      0485 6                                  .message_vect = dbg$nmake_arg_vect(
355      0486 6                                      dbg$noscope, 1, .dbg$src_next_lnum);
356      0487 6
357      0488 6

```

```

: 358 0489 6 RETURN sts$k_severe;
: 359 0490 6
: 360 0491 6
: 361 0492 6
: 362 0493 6
: 363 0494 6
: 364 0495 6
: 365 0496 6
: 366 0497 6
: 367 0498 6
: 368 0499 6
: 369 0500 6
: 370 0501 6 ! read the low line num
: 371 0502 6
: 372 0503 6 IF NOT dbg$nsave_decimal_integer(.input_desc, low_lnum,
: 373 0504 6 .message_vect)
: 374 0505 6 THEN
: 375 0506 6 RETURN sts$k_severe;
: 376 0507 6
: 377 0508 6 ! Now look for colon which signifies that the user has also
: 378 0509 6 ! specified a high line num.
: 379 0510 6
: 380 0511 6 IF dbg$nmatch(.input_desc,dbg$cs_colon,1)
: 381 0512 6 THEN
: 382 0513 6
: 383 0514 6 BEGIN
: 384 0515 6
: 385 0516 6 ! Get high line num
: 386 0517 6
: 387 0518 6 IF NOT dbg$nsave_decimal_integer(
: 388 0519 6 .input_desc,high_lnum, .message_vect)
: 389 0520 6 THEN
: 390 0521 6 RETURN sts$k_severe;
: 391 0522 6
: 392 0523 6 END
: 393 0524 6
: 394 0525 6 ELSE
: 395 0526 6
: 396 0527 6 ! high line num same as low line num
: 397 0528 6
: 398 0529 6 high_lnum = .low_lnum;
: 399 0530 6
: 400 0531 6 ! Fill in the fields of noun_node
: 401 0532 6
: 402 0533 6 noun_node[dbg$l_noun_value] = .low_lnum;
: 403 0534 6 noun_node[dbg$l_noun_value2] = .high_lnum;
: 404 0535 6
: 405 0536 6 ! Link in the noun node
: 406 0537 6
: 407 0538 6 IF NOT .first_flag
: 408 0539 6 ! The first noun has already been linked to the verb
: 409 0540 6 THEN
: 410 0541 6
: 411 0542 6 BEGIN
: 412 0543 6
: 413 0544 6 prev_noun_node[dbg$l_noun_link] = .noun_node;
: 414 0545 6 noun_node[dbg$l_noun_link] = 0;

```



```

DBG$CS_CR= P.AAD
DBG$CS_BACKSLASH= P.AAE
.EXTRN DBG$STA_GETSOURCEMOD
.EXTRN DBG$STA_SYMNAME
.EXTRN DBG$PRINT, DBG$FLUSHBUF
.EXTRN DBG$NNEXT_WORD, DBG$NEWLINE
.EXTRN DBG$NMATCH, DBG$GET_TEMPMEM
.EXTRN DBG$NMAKE_ARG_VECT
.EXTRN DBG$NSAVE_DECIMAL_INTEGER
.EXTRN DBG$NSYNTAX_ERROR
.EXTRN DBG$SCR_SCREEN_NORMAL
.EXTRN DBG$SRC_LINE_TO_REC
.EXTRN DBG$SRC_TYPE_LNUM_SOURCE
.EXTRN LIB$DO_COMMAND, LIB$SPAWN
.EXTRN SMGSSET_KEYPAD_MODE
.EXTRN SYSS$FAO, DBG$GB_KEYPAD_INPUT
.EXTRN DBG$GL_KEYBOARD_ID
.EXTRN DBG$GV_CONTROL, DBG$GL_DEVELOPER
.EXTRN DBG$GL_SCREEN_SOURCE
.EXTRN DBG$SRC_NEXT_MODRSTPTR
.EXTRN DBG$SRC_NEXT_LNUM

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

.ENTRY DBG$NPARSE_TYPE, Save R2,R3,R4,R5,R6,R7,R8,-; 0261
R9,R10,R11
MOVAB DBG$CS_CR, R11
MOVAB NAME_BUF, R10
SUBL2 #8, SP
PUSHL #4
CALLS #1, DBG$GET_TEMPMEM
ROVL RO, NOUN_NODE
MOVAB VERB_NODE, RO
MOVAB NOUN_NODE, 8(RO)
MOVAB #1, FIRST_FLAG
CLRAB MODRSTPTR
PUSHL #1
PUSHL R11
MOVAB INPUT_DESC, R4
PUSHL R4
CALLS #3, DBG$NMATCH
BLBC RO, 2$
TSTL DBG$SRC_NEXT_MODRSTPTR
1$
PUSHL #167136
BRB 5$
MOVAB DBG$SRC_NEXT_MODRSTPTR, 4(NOUN_NODE)
MOVAB DBG$SRC_NEXT_LNUM, RO
MOVAB RO, (NOUN_NODE)
MOVAB RO, 12(NOUN_NODE)
CLRAB 8(NOUN_NODE)
BRW 19$
CLRB NAME_BUF
MOVAB 4(R4), STRING_PTR
MOVZWL (R4), LENGTH
TSTL LENGTH
BLEQ 4$

```

```

OFFC 00000
5B 00000000' EF 9E 00002
5A 00000000' EF 9E 00009
5E 08 00010
00000000G 00 04 DD 00013
52 01 FB 00015
50 08 AC D0 0001F
08 A0 52 D0 00023
59 01 D0 00027
57 D4 0002A
01 DD 0002C
58 DD 0002E
54 04 AC D0 00030
00000000G 00 54 DD 00034
2C 03 FB 00036
00000000G 00 50 E9 0003D
00028CE0 08 12 00046
04 A2 00000000G 00 D0 00050 1$:
50 00000000G 00 D0 00058
62 50 D0 0005F
0C A2 50 D0 00062
08 A2 D4 00066
0157 31 00069
56 04 6A 94 0006C 2$:
55 64 D0 0006E
64 3C 00072
55 D5 00075 3$:
0A 15 00077

```

```

:
:
: 0325
:
: 0326
:
: 0328
: 0329
: 0333
:
:
:
:
: 0340
:
: 0344
:
: 0351
: 0355
:
: 0356
: 0360
: 0333
: 0391
: 0392
: 0393
: 0397
:

```

53		86	9A	00079	MOVZBL	(STRING_PTR)+, CHAR	0400	
		55	D7	0007C	DECL	LENGTH	0401	
20		53	D1	0007E	CPL	CHAR, #32	0402	
		F2	13	00081	BEQL	3\$		
		55	D5	00083	4\$: TSTL	LENGTH	0409	
		15	12	00085	BNEQ	6\$		
20		53	D1	00087	CPL	CHAR, #32		
		10	12	0008A	BNEQ	6\$		
	000280D0	8F	DD	0008C	PUSHL	#164048	0412	
0000000G	00	01	FB	00092	5\$: CALLS	#1, DBG\$NMAKE_ARG_VECT		
		01	10	31	00099	BRW	16\$	
		55	D5	0009C	6\$: TSTL	LENGTH	0419	
		28	15	0009E	BLEQ	7\$		
0000005C	8F	53	D1	000A0	CPL	CHAR, #92	0422	
		1F	13	000A7	BEQL	7\$		
	2C	53	D1	000A9	CPL	CHAR, #44		
		1A	13	000AC	BEQL	7\$		
	3A	53	D1	000AE	CPL	CHAR, #58	0423	
		15	13	000B1	BEQL	7\$		
	20	53	D1	000B3	CPL	CHAR, #32		
		10	13	000B6	BEQL	7\$		
		6A	96	000B8	INCB	NAME_BUF	0426	
	50	6A	9A	000BA	MOVZBL	NAME_BUF, R0	0427	
	6A40	53	90	000BD	MOVB	CHAR, NAME_BUF[R0]		
	53	86	9A	000C1	MOVZBL	(STRING_PTR)+, CHAR	0428	
		55	D7	000C4	DECL	LENGTH	0429	
		D4	11	000C6	BRB	6\$	0419	
0000005C	8F	53	D1	000C8	7\$: CPL	CHAR, #92	0432	
		28	12	000CF	BNEQ	9\$		
	04	56	D0	000D1	MOVL	STRING_PTR, 4(R4)	0437	
	64	55	B0	000D5	MOVW	LENGTH, (R4)	0438	
		5A	DD	000D8	PUSHL	R10	0444	
0000000G	00	01	FB	000DA	CALLS	#1, DBG\$STA_GETSOURCEMOD		
	04	50	D0	000E1	MOVL	R0, 4(NOUN_NODE)		
		0C	12	000E5	BNEQ	8\$	0449	
		5A	DD	000E7	PUSHL	R10	0452	
		01	DD	000E9	PUSHL	#1		
	000281E8	8F	DD	000EB	PUSHL	#164328		
		26	11	000F1	BRB	10\$		
	57	A2	D0	000F3	8\$: MOVL	4(NOUN_NODE), MODRSTPTR	0458	
		2E	11	000F7	BRB	12\$	0432	
		57	D5	000F9	9\$: TSTL	MODRSTPTR	0466	
		26	12	000FB	BNEQ	11\$		
		7E	D4	000FD	CLRL	-(SP)	0477	
0000000G	00	01	FB	000FF	CALLS	#1, DBG\$STA_GETSOURCEMOD		
	57	50	D0	00106	MOVL	R0, MODRSTPTR		
		18	12	00109	BNEQ	11\$	0479	
		00	DD	0010B	PUSHL	DBG\$SRC_NEXT_LNUM	0488	
	0000000G	01	DD	00111	PUSHL	#1	0487	
		8F	DD	00113	PUSHL	#166258		
0000000G	00	03	FB	00119	10\$: CALLS	#3, DBG\$NMAKE_ARG_VECT		
		00	89	31	00120	BRW	16\$	
	04	A2	57	D0	00123	11\$: MOVL	MODRSTPTR, 4(NOUN_NODE)	0496
		0C	AC	DD	00127	12\$: PUSHL	MESSAGE_VECT	0504
		04	AE	9F	0012A	PUSHAB	LOW_LNUM	0503
		54	DD	0012D	PUSHL	R4		
0000000G	00	03	FB	0012F	CALLS	#3, DBG\$NSAVE_DECIMAL_INTEGER		

	77		50	E9	00136	BLBC	R0, 17\$		
			01	DD	00139	PUSHL	#1		0511
		FA	AB	9F	0013B	PUSHAB	DBG\$CS_COLON		
			54	DD	0013E	PUSHL	R4		
00000000G	00		03	FB	00140	CALLS	#3, DBG\$NMATCH		
	14		50	E9	00147	BLBC	R0, 13\$		
		OC	AC	DD	0014A	PUSHL	MESSAGE_VECT		0519
		08	AE	9F	0014D	PUSHAB	HIGH_LNOM		0518
			54	DD	00150	PUSHL	R4		0519
00000000G	00		03	FB	00152	CALLS	#3, DBG\$NSAVE_DECIMAL_INTEGER		
	06		50	E8	00159	BLBS	R0, 14\$		
			52	11	0015C	BRB	17\$		0521
04	AE		6E	DD	0015E	MOVL	LOW_LNUM, HIGH_LNUM		0529
	62		6E	DD	00162	MOVL	LOW_LNUM, (NOUN_NODE)		0533
		04	AE	DD	00165	MOVL	HIGH_LNUM, 12(NOUN_NODE)		0534
			59	E8	0016A	BLBS	FIRST_FLAG, 15\$		0538
		08	AB	DD	0016D	MOVL	NOUN_NODE, 8(PREV_NOUN_NODE)		0544
			A2	D4	00171	CLRL	8(NOUN_NODE)		0545
			59	D4	00174	CLRL	FIRST_FLAG		0550
	58		52	DD	00176	MOVL	NOUN_NODE, PREV_NOUN_NODE		0551
			01	DD	00179	PUSHL	#1		0555
00000000G	00	0810	8F	BB	0017B	PUSHR	#^M<R4, R11>		
	3A		03	FB	0017F	CALLS	#3, DBG\$NMATCH		
			50	E8	00186	BLBS	R0, 19\$		
			01	DD	00189	PUSHL	#1		0562
		FE	AB	9F	0018B	PUSHAB	DBG\$CS_COMMA		
			54	DD	0018E	PUSHL	R4		
00000000G	00		03	FB	00190	CALLS	#3, DBG\$NMATCH		
	1A		50	E8	00197	BLBS	R0, 18\$		
			54	DD	0019A	PUSHL	R4		0568
00000000G	00		01	FB	0019C	CALLS	#1, DBG\$NNEXT_WORD		
			50	DD	001A3	PUSHL	R0		
00000000G	J0		01	FB	001A5	CALLS	#1, DBG\$NSYNTAX_ERROR		
	OC		50	DD	001AC	MOVL	R0, @MESSAGE_VECT		
			04	DD	001B0	MOVL	#4, R0		0569
			04	DD	001B3	RET			
			04	DD	001B4	PUSHL	#4		0574
00000000G	00		01	FB	001B6	CALLS	#1, DBG\$GET_TEMPMEM		
	52		50	DD	001BD	MOVL	R0, NOUN_NODE		
		FEA9	31	001C0	BRW	2\$			0367
			01	DD	001C3	MOVL	#1, R0		0577
	50		04	001C6	RET				0579

; Routine Size: 455 bytes, Routine Base: DBG\$CODE + 0000

: 449 0580 1
: 450 0581 1

```

452 0582 1 GLOBAL ROUTINE DBG$NEXECUTE_TYPE(VERB_NODE, MESSAGE_VECT) =
453 0583 1
454 0584 1 Functional Description
455 0585 1
456 0586 1 This routine performs the action associated with the TYPE
457 0587 1 command.
458 0588 1
459 0589 1 Formal Parameters
460 0590 1
461 0591 1 VERB_NODE - A longword containing the address of the
462 0592 1 head (verb) node.
463 0593 1
464 0594 1 MESSAGE_VECT - The address of a longword to contain the
465 0595 1 address of an error message vector
466 0596 1
467 0597 1 Implicit Inputs
468 0598 1
469 0599 1 The command tree contains a verb node and a linked list
470 0600 1 of one or more noun nodes.
471 0601 1
472 0602 1 Implicit Outputs
473 0603 1
474 0604 1 This routine calls a routine in DBG$SOURCE which displays the
475 0605 1 source lines to the user.
476 0606 1
477 0607 1 Routine Value
478 0608 1
479 0609 1 A completion code.
480 0610 1
481 0611 1 Completion Codes
482 0612 1
483 0613 1 ST$K_SUCCESS (1) - Success. Command executed
484 0614 1 ST$K_SEVERE (4) - Failure. The command could not be
485 0615 1 executed. An error message is constructed.
486 0616 1
487 0617 1 Side Effects
488 0618 1
489 0619 1 None
490 0620 1
491 0621 1
492 0622 2 BEGIN
493 0623 2
494 0624 2 MAP
495 0625 2 verb_node : REF dbg$verb_node;
496 0626 2
497 0627 2 LOCAL
498 0628 2 noun_node : REF dbg$noun_node,
499 0629 2 modnameptr;
500 0630 2
501 0631 2 noun_node = .verb_node[dbg$l_verb_object_ptr];
502 0632 2
503 0633 2 DO
504 0634 3 BEGIN
505 0635 3
506 0636 3 ! loop through linked list of noun nodes.
507 0637 3
508 0638 3 ! First output the module name.

```


		7E	7C	0003E	2\$:	CLRQ	-(SP)	:	0660
		7E	D4	00040		CLRL	-(SP)	:	
		0C	A2	DD	00042	PUSHL	12(NOUN_NODE)	:	0662
			7E	D4	00045	CLRL	-(SP)	:	0660
			62	DD	00047	PUSHL	(NOUN_NODE)	:	0661
		04	A2	DD	00049	PUSHL	4(NOUN_NODE)	:	0660
00000000G	00		07	FB	0004C	CALLS	#7, DBG\$SRC TYPE_LNUM_SOURCE	:	
	52	08	A2	D0	00053	MOVL	8(NOUN_NODE), NOON_NODE	:	0666
			B4	12	00057	BNEQ	1\$:	
	50		01	D0	00059	MOVL	#1, R0	:	0668
			04	0005C		RET		:	0670

; Routine Size: 93 bytes, Routine Base: DBG\$CODE + 01C7

```

: 542 0671 1 GLOBAL ROUTINE DBG$NPARSE_EDIT (INPUT_DESC, VERB_NODE, MESSAGE_VECT) =
: 543 0672 1
: 544 0673 1 Functional Description
: 545 0674 1
: 546 0675 1 Parse routine for the EDIT command. EDIT is the DEBUG interface
: 547 0676 1 to the EDITH editor.
: 548 0677 1
: 549 0678 1 The routine DBG$NPARSE_CMD in DBGNPARSE has already recognized
: 550 0679 1 the 'EDIT' verb and has called this routine to parse the
: 551 0680 1 rest of the command.
: 552 0681 1
: 553 0682 1 This routine takes a verb node for the EDIT verb, and a string
: 554 0683 1 descriptor for the remaining (unparsed) input.
: 555 0684 1
: 556 0685 1 The rest of the command execution tree is constructed.
: 557 0686 1
: 558 0687 1 Formal Parameters
: 559 0688 1
: 560 0689 1 INPUT_DESC - A longword containing the address of the
: 561 0690 1 command input descriptor.
: 562 0691 1
: 563 0692 1 VERB_NODE - A longword containing the address of the verb node
: 564 0693 1
: 565 0694 1 MESSAGE_VECT - The address of a longword to contain the address
: 566 0695 1 of a standard message argument vector.
: 567 0696 1
: 568 0697 1 Implicit Inputs
: 569 0698 1
: 570 0699 1 None
: 571 0700 1
: 572 0701 1 Implicit Outputs
: 573 0702 1
: 574 0703 1 On success, the command execution tree is constructed.
: 575 0704 1 On failure, a message argument vector is constructed or obtained.
: 576 0705 1
: 577 0706 1 Routine value
: 578 0707 1
: 579 0708 1 ST$K_SUCCESS (1) - Success. Command execution tree constructed.
: 580 0709 1 ST$K_SEVERE (4) - Failure. Error encountered. Message argument
: 581 0710 1 constructed and returned.
: 582 0711 1
: 583 0712 1 Side Effects
: 584 0713 1
: 585 0714 1 None
: 586 0715 1
: 587 0716 1
: 588 0717 2 BEGIN
: 589 0718 2
: 590 0719 2 MAP
: 591 0720 2 INPUT_DESC : REF DBG$STG_DESC,
: 592 0721 2 VERB_NODE : REF DBG$VERB_NODE;
: 593 0722 2
: 594 0723 2 BIND
: 595 0724 2 DBG$CS_EXIT = UPLIT BYTE (4, 'EXIT'),
: 596 0725 2 DBG$CS_NOEXIT = UPLIT BYTE (6, 'NOEXIT'),
: 597 0726 2 DBG$CS_BACKSLASH = UPLIT BYTE (1, DBG$K_BACKSLASH),
: 598 0727 2 DBG$CS_CR = UPLIT BYTE (1, DBG$K_CAR_RETURN).
```

4E

4E
46

20
75

20
75

4E

```

599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655

```

```

                DBG$CS_SLASH      = UPLIT BYTE (1, DBG$K_SLASH);
LOCAL
  CHAR
  DLINE_PTR: REF DBG$DLINE_ENTRY,
  DROW,
  LENGTH,
  LINENO,
  LINUM,
  MODRSTPTR,
  NAME_BUF: VECTOR [81, BYTE],
  NOUN_NODE: REF DBG$NOUN_NODE,
  RBEG,
  RLEN,
  SRC_DISP: REF DBG$DISP_ENTRY,
  STRING_PTR;

! Create and link a noun node.
NOUN_NODE = DBG$GET TEMPMEM(DBG$K_NOUN_NODE_SIZE);
VERB_NODE[DBG$L_VERB_OBJECT_PTR] = .NOUN_NODE;

! Pick up optional /EXIT or /NOEXIT. /NOEXIT is the default.
VERB_NODE[DBG$B_VERB_COMPOSITE] = EDIT_NOEXIT;
WHILE DBG$NMATCH (.INPUT_DESC, DBG$CS_SLASH, 1) DO
  BEGIN
    SELECTONE TRUE OF
      SET
        [DBG$NMATCH (.INPUT_DESC, DBG$CS_EXIT, 1)]:
          VERB_NODE[DBG$B_VERB_COMPOSITE] = EDIT_EXIT;
        [DBG$NMATCH (.INPUT_DESC, DBG$CS_NOEXIT, 1)]:
          VERB_NODE[DBG$B_VERB_COMPOSITE] = EDIT_NOEXIT;
        [DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)]:
          SIGNAL(DBG$_NEEDMORE);
        [OTHERWISE]:
          SIGNAL(DBG$_CMDSYNERR, 1, DBG$NNEXT_WORD(.INPUT_DESC));
    TES;
  END;

! Initialize module and line number.
MODRSTPTR = 0;
LINENO = 0;

! Check for EDIT <cr>

```

```

: 656
: 657
: 658
: 659
: 660
: 661
: 662
: 663
: 664
: 665
: 666
: 667
: 668
: 669
: 670
: 671
: 672
: 673
: 674
: 675
: 676
: 677
: 678
: 679
: 680
: 681
: 682
: 683
: 684
: 685
: 686
: 687
: 688
: 689
: 690
: 691
: 692
: 693
: 694
: 695
: 696
: 697
: 698
: 699
: 700
: 701
: 702
: 703
: 704
: 705
: 706
: 707
: 708
: 709
: 710
: 711
: 712

```

```

0785
0786
0787
0788
0789
0790
0791
0792
0793
0794
0795
0796
0797
0798
0799
0800
0801
0802
0803
0804
0805
0806
0807
0808
0809
0810
0811
0812
0813
0814
0815
0816
0817
0818
0819
0820
0821
0822
0823
0824
0825
0826
0827
0828
0829
0830
0831
0832
0833
0834
0835
0836
0837
0838
0839
0840
0841

```

```

IF DBG$NMATCH(.INPUT_DESC, DBG$CS_CR, 1)
THEN
  BEGIN
    ! Try to get default module and line number. If screen mode is set,
    ! then get this info from the source display on the screen. Otherwise,
    ! use defaulting rules based on the most recent TYPE command. If we
    ! are unable to default module name and line number then leave
    ! zeros in the variables to indicate this.
    SRC_DISP = .DBG$GL_SCREEN_SOURCE;
    IF .SRC_DISP NEQ 0
    THEN
      BEGIN
        MODRSTPTR = .SRC_DISP[DBG$L_DISP_MODPTR];

        ! Do the computation to figure out what line number is
        ! the center of the source display.
        RBEG = .SRC_DISP[DBG$W_DISP_RBEG];
        RLEN = .SRC_DISP[DBG$W_DISP_RLEN];
        DROW = .SRC_DISP[DBG$W_DISP_DROW];
        DLINE_PTR = .SRC_DISP[DBG$L_DISP_START_LINE_PTR];
        LINUM = .DLINE_PTR[DBG$L_DLINE_LINUM];
        LINENO = (.DROW-.RBEG) + .RLEN/2 + .LINUM;
        END
      ELSE IF .DBG$SRC_NEXT_MODRSTPTR NEQ 0
      THEN
        BEGIN
          MODRSTPTR = .DBG$SRC_NEXT_MODRSTPTR;
          LINENO = .DBG$SRC_NEXT_LNOM - 1;
          END;
        END
      ELSE
        BEGIN
          ! Initialize the buffer which will hold the module name.
          ! Set up to scan the input.
          name_buf[0] = 0;
          string_ptr = .input_desc[dsc$a_pointer];
          length = .input_desc[dsc$w_length];

          ! Read past leading blanks
          WHILE .length GTR 0 DO
            BEGIN
              char = ch$rchar a(string_ptr);
              length = .length - 1;
              IF .char NEQ dbg$blank

```

```

: 713      0842 4
: 714      0843 4
: 715      0844 4
: 716      0845 4
: 717      0846 4
: 718      0847 4
: 719      0848 4
: 720      0849 4
: 721      0850 4
: 722      0851 4
: 723      0852 4
: 724      0853 4
: 725      0854 4
: 726      0855 4
: 727      0856 4
: 728      0857 4
: 729      0858 4
: 730      0859 4
: 731      0860 4
: 732      0861 4
: 733      0862 4
: 734      0863 4
: 735      0864 4
: 736      0865 4
: 737      0866 4
: 738      0867 4
: 739      0868 4
: 740      0869 4
: 741      0870 4
: 742      0871 4
: 743      0872 4
: 744      0873 4
: 745      0874 4
: 746      0875 4
: 747      0876 4
: 748      0877 4
: 749      0878 4
: 750      0879 4
: 751      0880 4
: 752      0881 4
: 753      0882 4
: 754      0883 4
: 755      0884 4
: 756      0885 4
: 757      0886 4
: 758      0887 4
: 759      0888 4
: 760      0889 4
: 761      0890 4
: 762      0891 4
: 763      0892 4
: 764      0893 4
: 765      0894 4
: 766      0895 4
: 767      0896 4
: 768      0897 4
: 769      0898 4

      THEN
      EXITLOOP;
      END;

      ! If the length reaches zero then it is an error.
      IF .length EQL 0
      THEN
        $DBG_ERROR('DBGNTYPE\DBG$NPARSE_EDIT');

      ! Read until we reach a separating character.
      ! Place the characters into name_buf as we read them.
      WHILE .length GTR 0 DO
      BEGIN
        IF .char EQL '\' OR .char EQL ' '
        THEN
          EXITLOOP;
          name_buf[0] = .name_buf[0] + 1;
          IF .name_buf[0] GTR 80
          THEN
            SIGNAL(dbg$_cmdsynerr, 1, dbg$next_word(.input_desc));
            name_buf[.name_buf[0]] = .char;
            char = ch$rchar_a(string_ptr);
            length = .length - 1;
          END;

      ! Backslash signifies that we just read a module name.
      IF .char EQL '\'
      THEN
      BEGIN
        ! Update the input descriptor
        input_desc[dsc$a_pointer] = .string_ptr;
        input_desc[dsc$w_length] = .length;

        ! Convert the name to an rst pointer.
        modrstptr = dbg$sta_getsourcemod(name_buf);

        ! If the above routine returned zero then the user entered
        ! an invalid module.
        IF .modrstptr EQL 0
        THEN
          SIGNAL(dbg$_nosuchmodu, 1, name_buf);
        END
      ELSE

```

```

: 770      0899  4      BEGIN
: 771      0900  4
: 772      0901  4
: 773      0902  4      ! No module was specified, so try to fill
: 774      0903  4      ! in a module based on (1) module in current
: 775      0904  4      ! source display, or (2) module given by previous
: 776      0905  4      ! TYPE command, or (3) current scope
: 777      0906  4
: 778      0907  4      SRC_DISP = .DBG$GL_SCREEN_SOURCE;
: 779      0908  4      IF .SRC_DISP NEQ 0
: 780      0909  4      THEN
: 781      0910  4          MODRSTPTR = .SRC_DISP[DBG$L_DISP_MODPTR]
: 782      0911  4
: 783      0912  4      ELSE IF .DBG$SRC_NEXT_MODRSTPTR NEQ 0
: 784      0913  4      THEN
: 785      0914  4          MODRSTPTR = .DBG$SRC_NEXT_MODRSTPTR
: 786      0915  4
: 787      0916  4      ELSE
: 788      0917  4          modrstptr = dbg$sta_getsourcemod(0);
: 789      0918  4      END;
: 790      0919  3
: 791      0920  3
: 792      0921  3      ! Read the line number.
: 793      0922  3
: 794      0923  3      IF NOT dbg$nsave_decimal_integer(.input_desc, lineno, .message_vect)
: 795      0924  3      THEN
: 796      0925  3          RETURN sts$k_severe;
: 797      0926  2      END;
: 798      0927  2
: 799      0928  2
: 800      0929  2      ! Fill in the fields of the noun node with the module and
: 801      0930  2      ! line number.
: 802      0931  2
: 803      0932  2      NOUN_NODE[DBG$L_ADJECTIVE_PTR] = .MODRSTPTR;
: 804      0933  2      NOUN_NODE[DBG$L_NOUN_VALUE] = .LINENO;
: 805      0934  2
: 806      0935  2
: 807      0936  2      ! We have finished parsing the EDIT command. Return success.
: 808      0937  2
: 809      0938  2      RETURN STS$K_SUCCESS;
: 810      0939  1      END;

```

													.PSECT		DBG\$PLIT,NOWRT, SHR, PIC,0						
													04	00015	P.AAG:	.BYTE	4				
													54	49	58	45	00016	.ASCII	\EXIT\		
														06	0001A	P.AAH:	.BYTE	6			
													54	49	58	45	4F	4E	0001B	.ASCII	\NOEXIT\
														5C	01	00021	P.AAI:	.BYTE	1, 92		
														0D	01	00023	P.AAJ:	.BYTE	1, 13		
														2F	01	00025	P.AAK:	.BYTE	1, 47		
4E	24	47	42	44	5C	45	50	59	54	4E	47	42	44	18	00027	P.AAL:	.ASCII	<24>\DBGNTYPE\<>92>\DBG\$NPARSE_EDIT\			
					54	49	44	45	5F	45	53	52	41	50	00036						

DBG\$CS_EXIT= P.AAG

DBG\$CS_NOEXIT= P.AAH
DBG\$CS_BACKSLASH= P.AAI
DBG\$CS_CR= P.AAJ
DBG\$CS_SLASH= P.AAK

				.PSECT	DBG\$CODE,NOWRT,	SHR,	PIC,0	
				OFFC	00000			
				.ENTRY	DBG\$NPARSE_EDIT,	Save R2,R3,R4,R5,R6,R7,R8,-;		0671
5B	00000000G	00	9E	00002	MOVAB	R9,R10,R11		
5A	00000000'	EF	9E	00009	MOVAB	DBG\$NMATCH, R11		
5E	A8	AE	9E	00010	MOVAB	DBG\$CS_CR, R10		
		04	DD	00014	PUSHL	#4		0748
00000000G		01	FB	00016	CALLS	#1, DBG\$GET_TEMPMEM		
		50	DD	0001D	MOVL	R0, NOUN_NODE		
		52	08	AC	DD	00020		0749
08	A2	59	DD	00024	MOVL	VERB_NODE, R2		
01	A2	02	90	00028	MOVL	NOUN_NODE, 8(R2)		
		54	04	AC	DD	0002C		0754
		01	DD	00030	1\$:	PUSHL	#1	0755
		02	AA	9F	00032	PUSHAB	DBG\$CS_SLASH	
		54	DD	00035	PUSHL	R4		
6B		03	FB	00037	CALLS	#3, DBG\$NMATCH		
63		50	E9	0003A	BLBC	R0, 5\$		
		01	DD	0003D	PUSHL	#1		0761
		F2	AA	9F	0003F	PUSHAB	DBG\$CS_EXIT	
		54	DD	00042	PUSHL	R4		
6B		03	FB	00044	CALLS	#3, DBG\$NMATCH		
01	01	50	D1	00047	CMPL	R0, #1		
		06	12	0004A	BNEQ	2\$		
		01	90	0004C	MOVB	#1, 1(R2)		0762
		DE	11	00050	BRB	1\$		
		01	DD	00052	2\$:	PUSHL	#1	0764
		F7	AA	9F	00054	PUSHAB	DBG\$CS_NOEXIT	
		54	DD	00057	PUSHL	R4		
6B		03	FB	00059	CALLS	#3, DBG\$NMATCH		
01	01	50	D1	0005C	CMPL	R0, #1		
		06	12	0005F	BNEQ	3\$		
		02	90	00061	MOVB	#2, 1(R2)		0765
		C9	11	00065	BRB	1\$		
		01	DD	00067	3\$:	PUSHL	#1	0767
		0410	8F	BB	00069	PUSHR	#M<R4,R10>	
6B		03	FB	0006D	CALLS	#3, DBG\$NMATCH		
01		50	D1	00070	CMPL	R0, #1		
		0F	12	00073	BNEQ	4\$		
00000000G	00	8F	DD	00075	PUSHL	#164048		0768
		01	FB	0007B	CALLS	#1, LIB\$SIGNAL		
		AC	11	00082	BRB	1\$		
00000000G	00	54	DD	00084	4\$:	PUSHL	R4	0771
		01	FB	00086	CALLS	#1, DBG\$NNEXT_WORD		
		50	DD	0008D	PUSHL	R0		
		01	DD	0008F	PUSHL	#1		
00000000G	00	8F	DD	00091	PUSHL	#167608		
		03	FB	00097	CALLS	#3, LIB\$SIGNAL		
		90	11	0009E	BRB	1\$		0755
		56	D4	000A0	5\$:	CLRL	MODRSTPTR	0779

0000005C	8F		BB 11 00168	BRB 11\$: 0857
			55 D1 0016A 13\$:	CMPL CHAR, #92	: 0874
04	A4		2A 12 00171	BNEQ 14\$	
	64		58 D0 00173	MOVL STRING_PTR, 4(R4)	: 0881
		04	57 B0 00177	MOVW LENGTH, (R4)	: 0882
00000000G	00		AE 9F 0017A	PUSHAB NAME_BUF	: 0887
	56		01 FB 0017D	CALLS #1, DBG\$STA_GETSOURCEMOD	
			50 D0 00184	MOVL R0, MODRSTPTR	
		04	38 12 00187	BNEQ 17\$: 0893
			AE 9F 00189	PUSHAB NAME_BUF	: 0895
			01 DD 0018C	PUSHL #1	
00000000G	00	000281E8	8F DD 0018E	PUSHL #164328	
			03 FB 00194	CALLS #3, LIB\$SIGNAL	
			24 11 0019B	BRB 17\$: 0874
	52	00000000G	00 D0 0019D 14\$:	MOVL DBG\$GL_SCREEN_SOURCE, SRC_DISP	: 0907
			06 13 001A4	BEQL 15\$: 0908
	56	34	A2 D0 001A6	MOVL 52(SRC_DISP), MODRSTPTR	: 0910
			15 11 001AA	BRB 17\$	
	50	00000000G	00 D0 001AC 15\$:	MOVL DBG\$SRC_NEXT_MODRSTPTR, R0	: 0912
			09 12 001B3	BNEQ 16\$	
			7E D4 001B5	CLRL -(SP)	: 0917
00000000G	00		01 FB 001B7	CALLS #1, DBG\$STA_GETSOURCEMOD	
	56		50 D0 001BE 16\$:	MOVL R0, MODRSTPTR	
		0C	AC DD 001C1 17\$:	PUSHL MESSAGE_VECT	: 0923
		04	AE 9F 001C4	PUSHAB LINENO	
			54 DD 001C7	PUSHL R4	
00000000G	00		03 FB 001C9	CALLS #3, DBG\$NSAVE_DECIMAL_INTEGER	
	04		50 E8 001D0	BLBS R0, 18\$	
	50		04 D0 001D3	MOVL #4, R0	: 0925
			04 001D6	RET	
	04	A9	56 D0 001D7 18\$:	MOVL MODRSTPTR, 4(NOUN_NODE)	: 0932
		69	6E D0 001DB	MOVL LINENO, (NOUN_NODE)	: 0933
		50	01 D0 001DE	MOVL #1, R0	: 0938
			04 001E1	RET	: 0939

: Routine Size: 482 bytes, Routine Base: DBG\$CODE + 0224

```

812 0940 1 GLOBAL ROUTINE DBG$NEXECUTE_EDIT (VERB_NODE, MESSAGE_VECT) =
813 0941 1
814 0942 1 Functional Description
815 0943 1
816 0944 1 This routine performs the action associated with the EDIT
817 0945 1 command. EDIT is the interface to the EDITH editor.
818 0946 1
819 0947 1 This routine constructs a DCL command line which invokes EDITH.
820 0948 1 Either LIB$SPAWN or LIB$DO_COMMAND is called to execute the
821 0949 1 command line.
822 0950 1
823 0951 1 Formal Parameters
824 0952 1
825 0953 1 VERB_NODE - A longword containing the address of the
826 0954 1 verb node of the command execution tree.
827 0955 1 This tree has already been built by the
828 0956 1 DBG$NPARSE_EDIT routine.
829 0957 1
830 0958 1 MESSAGE_VECT - The address of a longword to contain the
831 0959 1 address of an error message vector
832 0960 1
833 0961 1 Implicit Inputs
834 0962 1
835 0963 1 The command tree contains a verb node and a noun node.
836 0964 1
837 0965 1 Implicit Outputs
838 0966 1
839 0967 1 This routine constructs a command line and calls either
840 0968 1 LIB$SPAWN or LIB$DO_COMMAND to execute the command line.
841 0969 1
842 0970 1 Routine Value
843 0971 1
844 0972 1 A completion code.
845 0973 1
846 0974 1 Completion Codes
847 0975 1
848 0976 1 ST$K_SUCCESS (1) - Success. Command executed
849 0977 1 ST$K_SEVERE (4) - Failure. The command could not be
850 0978 1 executed. An error message is constructed.
851 0979 1
852 0980 1 Side Effects
853 0981 1
854 0982 1 If the EDIT/EXIT form of the command was given, the debugging
855 0983 1 session is terminated.
856 0984 1
857 0985 1
858 0986 2 BEGIN
859 0987 2
860 0988 2 MAP
861 0989 2 VERB_NODE : REF DBG$VERB_NODE;
862 0990 2
863 0991 2 LOCAL
864 0992 2 COMMAND_LINE: VECTOR[132,BYTE], ! Command line to invoke EDITH
865 0993 2 CTRLBUF
866 0994 2 CTRLDESC: DBG$STG_DESC,
867 0995 2 FABPTR: REF $FAB_DECL, ! FAB for searching for newer file ver.
868 0996 2 FAOBUF: VECTOR[80, BYTE],

```

```

869 0997 2 FAODESC: DBG$STG_DESC,
870 0998 2 FAOLEN,
871 0999 2 LENGTH, ! Length of command line
872 1000 2 LINENO, ! Line number given in EDIT command
873 1001 2 MODRSTPTR, ! Module given in EDIT command
874 1002 2 NAMESA, ! Ptr to filename buff for $PARSE/$SEARCH
875 1003 2 NAMPTR: REF $NAM_DECL,
876 1004 2 NAMRSA, ! Ptr to filename buff for $PARSE/$SEARCH
877 1005 2 NOUN_NODE : REF DBG$NOUN_NODE, ! Noun node in execution tree
878 1006 2 OLDNAMPTR : REF $NAM_DECL, ! NAM block of current source file
879 1007 2 PTR,
880 1008 2 RECNO, ! Record number in file
881 1009 2 STATUS, ! Return status temp.
882 1010 2 STG_DESC: DBG$STG_DESC; ! String descriptor
883 1011 2
884 1012 2
885 1013 2
886 1014 2 ! Recover the noun node. From this, obtain the module RST pointer and
887 1015 2 ! line number.
888 1016 2
889 1017 2 NOUN_NODE = .VERB_NODE[DBG$VERB_OBJECT_PTR];
890 1018 2 MODRSTPTR = .NOUN_NODE[DBG$ADJECTIVE_PTR];
891 1019 2 LINENO = .NOUN_NODE[DBG$NOUN_VALUE];
892 1020 2
893 1021 2
894 1022 2 ! Build the command line and the string descriptor
895 1023 2 ! with which we will invoke EDITH. We start by initializing
896 1024 2 ! the current length of the command line and a pointer into
897 1025 2 ! the command line.
898 1026 2
899 1027 2 LENGTH = 0;
900 1028 2 PTR = COMMAND_LINE;
901 1029 2
902 1030 2
903 1031 2 ! Append the LSEEDIT command to the line.
904 1032 2 ! Note - the command name may change. We leave a few extra
905 1033 2 ! spaces so we can patch in whatever command name they choose,
906 1034 2 ! up to 15 characters.
907 1035 2 ! Additional note - it now appears that the command name will
908 1036 2 ! be EDIT/LSE. However, this change will not happen until at least
909 1037 2 ! version 4.1 of VMS.
910 1038 2
911 1039 2 LENGTH = .LENGTH + 15;
912 1040 2 PTR = CH$MOVE(15, UPLIT BYTE(%ASCII 'LSEEDIT '), .PTR);
913 1041 2
914 1042 2
915 1043 2 ! The following block is only executed when a source file is present.
916 1044 2 ! Translate the module and line number into a NAM block pointer and record
917 1045 2 ! number within the file. This is done by calling a routine in DBG$SOURCE.
918 1046 2 ! Also, do the checking for a more recent version of the source file,
919 1047 2 ! and issue any appropriate messages. After this is done, add the
920 1048 2 ! file name and the appropriate /START_POSITION qualifier.
921 1049 2
922 1050 2 IF .MODRSTPTR NEQ 0
923 1051 2 THEN
924 1052 2 BEGIN
925 1053 2 DBG$SRC_LINE_TO_REC(.MODRSTPTR, .LINENO, OLDNAMPTR, RECNO);

```

```

926      1054      3
927      1055
928      1056      ! Get memory for the FAB and NAM blocks.
929      1057
930      1058      FABPTR = DBG$GET_TEMPMEM((FAB$C_BLN+3)/4);
931      1059      NAMPTR = DBG$GET_TEMPMEM((NAM$C_BLN+3)/4);
932      1060      NAMESA = DBG$GET_TEMPMEM((NAM$C_MAXRSS+3)/4);
933      1061      NAMRSA = DBG$GET_TEMPMEM((NAM$C_MAXRSS+3)/4);
934      1062
935      1063
936      1064      ! Initialize the FAB and NAM blocks for $PARSE and $SEARCH.
937      1065
938      1066      $FAB_INIT(FAB = .FABPTR,
939      1067      P      NAM = .NAMPTR,
940      1068      P      FNA = .OLDNAMPTR[NAM$L_ESA],
941      1069      P      FNS = (.OLDNAMPTR[NAM$B_ESL] - .OLDNAMPTR[NAM$B_VER]) );
942      1070
943      1071      $NAM_INIT(NAM = .NAMPTR,
944      1072      P      ESS = NAM$C_MAXRSS,
945      1073      P      ESA = .NAMESA,
946      1074      P      RSS = NAM$C_MAXRSS,
947      1075      RSA = .NAMRSA );
948      1076
949      1077      ! Parse the spec, and then search for the most recent version.
950      1078      ! The PARSE should never fail, since we are using a string from a
951      1079      ! previous file OPEN. However, under very unusual circumstances,
952      1080      ! the SEARCH may fail (if a user spawned out from DEBUG and renamed
953      1081      ! or deleted the file, for example).
954      1082
955      1083      STATUS = $PARSE(FAB = .FABPTR);
956      1084      IF NOT .STATUS
957      1085      THEN
958      1086          $DBG_ERROR('DBGNTYPE\DBG$NEXECUTE_EDIT - RMS Parse error',.STATUS);
959      1087
960      1088      STATUS = $SEARCH(FAB = .FABPTR);
961      1089      IF NOT .STATUS
962      1090      THEN
963      1091          SIGNAL(DBG$_EDITNOFILE,2,
964      1092          .OLDNAMPTR[NAM$B_ESL],.OLDNAMPTR[NAM$L_ESA],.STATUS)
965      1093
966      1094      ELSE
967      1095      BEGIN
968      1096          ! Issue a message about the file we are editing.
969      1097          !
970      1098          SIGNAL(DBG$_EDITFILE,2,.NAMPTR[NAM$B_RSL],.NAMPTR[NAM$L_RSA]);
971      1099
972      1100
973      1101          ! Now compare the result of the SEARCH with the old filename, and
974      1102          ! issue a pair of messages if the version differs.
975      1103
976      1104          IF CH$NEQ(.OLDNAMPTR[NAM$B_ESL], .OLDNAMPTR[NAM$L_ESA],
977      1105          .NAMPTR[NAM$B_RSL], .NAMPTR[NAM$L_RSA] )
978      1106
979      1107      THEN
980      1108          BEGIN
981      1109              SIGNAL(DBG$_EDITREVER);
982      1110              SIGNAL(DBG$_EDITDISVER,2,.OLDNAMPTR[NAM$B_ESL],.OLDNAMPTR[NAM$L_ESA]);
          END;

```

```

: 983 1111 4
: 984 1112 4
: 985 1113 4
: 986 1114 4
: 987 1115 4
: 988 1116 4
: 989 1117 4
: 990 1118 4
: 991 1119 4
: 992 1120 4
: 993 1121 4
: 994 1122 4
: 995 1123 4
: 996 1124 4
: 997 1125 4
: 998 1126 4
: 999 1127 4
1000 1128 4
1001 1129 4
1002 1130 4
1003 1131 4
1004 1132 4
1005 1133 4
1006 1134 4
1007 1135 4
1008 1136 4
1009 1137 4
1010 1138 4
1011 1139 4
1012 1140 4
1013 1141 4
1014 1142 4
1015 1143 4
1016 1144 4
1017 1145 4
1018 1146 4
1019 1147 4
1020 1148 4
1021 1149 4
1022 1150 4
1023 1151 4
1024 1152 4
1025 1153 3
1026 1154 3
1027 1155 2
1028 1156 2
1029 1157 2
1030 1158 2
1031 1159 2
1032 1160 2
1033 1161 2
1034 1162 2
1035 1163 2
1036 1164 2
1037 1165 2
1038 1166 2
: 1039 1167 2

! Append the start position qualifier. Again, leave some
! extra space for flexibility.
LENGTH = .LENGTH + 21;
PTR = CH$MOVE(21, UPLIT BYTE(%ASCII '/START_POSITION = '), .PTR);

! We need to call $FAO
! to translate the record number that we have into a
! string.
CTRLBUF = UPLIT BYTE(%ASCII '(!UL,1) ');
CTRLDESC[DSC$B_CLASS] = DSC$K_CLASS_S;
CTRLDESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
CTRLDESC[DSC$W_LENGTH] = 8;
CTRLDESC[DSC$A_POINTER] = .CTRLBUF;
FAOLEN = 0;
FAODESC[DSC$B_CLASS] = DSC$K_CLASS_S;
FAODESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
FAODESC[DSC$W_LENGTH] = 80;
FAODESC[DSC$A_POINTER] = FAOBUF;
IF NOT SYSS$FAO (CTRLDESC, FAOLEN, FAODESC, .RECNO)
THEN
  $DBG_ERROR('DBGNTYPE\DBG$NEXECUTE_EDIT $FAO failed');
LENGTH = .LENGTH + .FAOLEN;
IF .LENGTH GTR 132
THEN
  $DBG_ERROR('EDIT command: Command line buffer has overflowed');
PTR = CH$MOVE(.FAOLEN, FAOBUF, .PTR);

! Append the file name to the command line.
LENGTH = .LENGTH + .NAMPTR[NAM$B_RSL];
IF .LENGTH GTR 132
THEN
  $DBG_ERROR('EDIT command: Command line buffer has overflowed');
PTR = CH$MOVE(.NAMPTR[NAM$B_RSL], .NAMPTR[NAM$L_RSA], .PTR);
END
END

ELSE
  SIGNAL(DBG$_EDITNOFILE);

! Now build a string descriptor with the command line.
!
STG_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
STG_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
STG_DESC[DSC$W_LENGTH] = .LENGTH;
STG_DESC[DSC$A_POINTER] = COMMAND_LINE;

! Developer bit 1 causes us to print the result but not

```


Address	Disassembly	Comment	PC
0060 8F 34 A8 00	MOVW #20483, (FABPTR)		1075
	MOVB #2, 22(FABPTR)		
	MOVB #2, 31(FABPTR)		
	MOVL NAMPTR, 40(FABPTR)		
	MOVL OLDNAMPTR, R6		
	MOVL 12(R6), 44(FABPTR)		
	SUBB3 61(R6), 11(R6), 52(FABPTR)		
	MOVC5 #0, (SP), #0, #96, (NAMPTR)		
	MOVW #24578, (NAMPTR)		
	MNEGB #1, 2(NAMPTR)		
	MOVL NAMRSA, 4(NAMPTR)		
	MNEGB #1, 10(NAMPTR)		
	MOVL NAMESA, 12(NAMPTR)		
	PUSHL FABPTR		1083
	CALLS #1, SYSSPARSE		
	MOVL R0, STATUS		
	BLBS STATUS, 2\$		1084
	PUSHL STATUS		1086
	PUSHAB P.AAN		
	PUSHL #1		
	PUSHL #164706		
	CALLS #4, LIBSSIGNAL		
	PUSHL FABPTR		1088
	CALLS #1, SYSSSEARCH		
	MOVL R0, STATUS		
	BLBS STATUS, 3\$		1089
	PUSHL STATUS		1092
	PUSHL 12(R6)		
	MOVZBL 11(R6), -(SP)		
	PUSHL #2		1091
	PUSHL #167947		
	CALLS #5, LIBSSIGNAL		
	BRW 9\$		
	PUSHL 4(NAMPTR)		1098
	MOVZBL 3(NAMPTR), -(SP)		
	PUSHL #2		
	PUSHL #167915		
	CALLS #4, LIBSSIGNAL		
	MOVZBL 11(R6), R1		1104
	MOVZBL 3(NAMPTR), R0		1105
	CMPC5 R1, @12(R6), #0, R0, @4(NAMPTR)		1104
	BEQL 4\$		
	PUSHL #167923		1108
	CALLS #1, LIBSSIGNAL		
	PUSHL 12(R6)		1109
	MOVZBL 11(R6), -(SP)		
	PUSHL #2		
	PUSHL #167931		
	CALLS #4, LIBSSIGNAL		
	ADDL2 #21, LENGTH		1116
	MOVC3 #21, P.AAO, (PTR)		1117
	MOVL R3, PTR		
	MOVAB P.AAP, CTRLBUF		1124
	MOVL #17694728, CTRLDESC		1127

0084	CE		50	DO	00177	MOVL	CTRLBUF, CTRLDESC+4	1128
		10	AE	D4	0017C	CLRL	FAOLEN	1129
24	AE	010E0050	8F	DO	0017F	MOVL	#17694800, FAODESC	1132
28	AE			9E	00187	MOVAB	FAOBUF, FAODESC+4	1133
		08	AE	DD	0018C	PUSHL	RECNO	1134
		28	AE	9F	0018F	PUSHAB	FAODESC	
		18	AE	9F	00192	PUSHAB	FAOLEN	
		008C	CE	9F	00195	PUSHAB	CTRLDESC	
00000000G	00		04	FB	00199	CALLS	#4, SYSSFAO	
	15		50	EB	001A0	BLBS	RO, 5\$	
		00000000'	EF	9F	001A3	PUSHAB	P.AAQ	1136
		00028362	01	DD	001A9	PUSHL	#1	
00000000G	00		8F	DD	001AB	PUSHL	#164706	
	59	10	03	FB	001B1	CALLS	#3, LIBSSIGNAL	1137
00000084	8F		AE	CO	001B8	ADDL2	FAOLEN, LENGTH	1138
			59	D1	001BC	CMPL	LENGTH, #132	
		00000000'	15	15	001C3	BLEQ	6\$	
		00028362	EF	9F	001C5	PUSHAB	P.AAR	1140
			01	DD	001CB	PUSHL	#1	
00000000G	00		8F	DD	001CD	PUSHL	#164706	
6A	30	AE	03	FB	001D3	CALLS	#3, LIBSSIGNAL	
		10	AE	28	0C DA	6\$: MOV C3	FAOLEN, FAOBUF, (PTR)	1141
		5A	53	DO	0C1E0	MOVL	R3, PTR	
		50	A7	9A	001E3	MOVZBL	3(NAMPTR), RO	1146
		59	50	CO	001E7	ADDL2	RO, LENGTH	
00000084	8F		59	D1	001EA	CMPL	LENGTH, #132	1147
		00000000'	15	15	001F1	BLEQ	7\$	
		00028362	EF	9F	001F3	PUSHAB	P.AAS	1149
			01	DD	001F9	PUSHL	#1	
00000000G	00		8F	DD	001FB	PUSHL	#164706	
	50	03	03	FB	00201	CALLS	#3, LIBSSIGNAL	
6A	04	B7	A7	9A	00208	7\$: MOVZBL	3(NAMPTR), RO	1150
		5A	50	28	0020C	MOV C3	RO, #4(NAMPTR), (PTR)	
			53	DO	00211	MOVL	R3, PTR	
		0002900B	0D	11	00214	BRB	9\$	1089
00000000G	00		8F	DD	00216	8\$: PUSHL	#167947	1156
	1A	010E	01	FB	0021C	CALLS	#1, LIBSSIGNAL	
	18		8F	BO	00223	9\$: MOVW	#270, STG_DESC+2	1162
	1C	FF7C	59	BO	00229	MOVW	LENGTH, STG_DESC	1163
19	00000000G	00	CD	9E	0022D	MOVAB	COMMAND LINE, STG_DESC+4	1164
		18	01	E1	00233	BBC	#1, DBG\$GL_DEVELOPER, 10\$	1170
		00000000'	AE	9F	0023B	PUSHAB	STG_DESC	1173
			EF	9F	0023E	PUSHAB	P.AAT	
00000000G	00		02	FB	00244	CALLS	#2, DBG\$PRINT	
00000000G	00		00	FB	0024B	CALLS	#0, DBG\$NEWLINE	1174
			69	11	00252	BRB	15\$	1170
00000000G	00		00	FB	00254	10\$: CALLS	#0, DBG\$SCR_SCREEN_NORMAL	1182
	50	01	AB	9A	0025B	MOVZBL	1(R11), RO	1187
	01		50	91	0025F	CMPB	RO, #1	1190
			13	12	00262	BNEQ	11\$	
00000000G	00		10	88	00264	BISB2	#16, DBG\$GV_CONTROL	1192
		18	AE	9F	0026B	PUSHAB	STG_DESC	1193
00000000G	00		01	FB	0026E	CALLS	#1, LIB\$DO_COMMAND	
			46	11	00275	BRB	15\$	1187
	02		50	91	00277	11\$: CMPB	RO, #2	1196
			2C	12	0027A	BNEQ	14\$	
		18	AE	9F	0027C	PUSHAB	STG_DESC	1200

00000000G	00		01	FB	0027F		CALLS	#1, LIB\$SPAWN		
	06	00000000G	00	E9	00286		BLBC	DBG\$GB_KEYPAD_INPUT, 12\$:	1201
	14		01	D0	0028D		MOVL	#1, X	:	1202
			03	11	00291		BRB	13\$:	
			14	AE	D4 00293	12\$:	CLRL	X	:	1203
			14	AE	9F 00296	13\$:	PUSHAB	X	:	1204
00000000G	00	00000000G	00	9F	00299		PUSHAB	DBG\$GL_KEYBOARD_ID	:	
			02	FB	0029F		CALLS	#2, SMG\$SET_KEYPAD_MODE	:	
			15	11	002A6		BRB	15\$:	1187
		00000000'	EF	9F	002A8	14\$:	PUSHAB	P.AAU	:	1208
			01	DD	002AE		PUSHL	#1	:	
		00028362	8F	DD	002B0		PUSHL	#164706	:	
00000000G	00		03	FB	002B6		CALLS	#3, LIB\$SIGNAL	:	
	50		01	D0	002BD	15\$:	MOVL	#1, R0	:	1215
			04	002C0			RET		:	1216

: Routine Size: 705 bytes, Routine Base: DBG\$CODE + 0406

: 1089 1217 0 END ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
DBG\$PLIT	321	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(0)
DBG\$OWN	81	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
DBG\$CODE	1735	NOVEC, NOWRT, RD, EXL, 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	68	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	84	5	97	00:02.1
-\$255\$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1	418	0	0	31	00:00.3
-\$255\$DUA28:[DEBUG.OBJ]DBGMSG.L32;1	386	10	2	22	00:00.3
-\$255\$DUA28:[DEBUG.OBJ]DBGGEN.L32;1	150	0	0	12	00:00.3

: Information: 1
: Warnings: 0
: Errors: 0

COMMAND QUALIFIERS

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

: Size: 1735 code + 402 data bytes
: Run Time: 00:34.6
: Elapsed Time: 01:35.8
: Lines/CPU Min: 2109
: Lexemes/CPU-Min: 15119
: Memory Used: 252 pages
: Compilation Complete

