

CCCCCCCCCCCC	LLL	IIIIIIII	UUU	UUU	TTTTTTTTTTTTTTTT	LLL
CCCCCCCCCCCC	LLL	IIIIIIII	UUU	UUU	TTTTTTTTTTTTTTTT	LLL
CCCCCCCCCCCC	LLL	IIIIIIII	UUU	UUU	TTTTTTTTTTTTTTTT	LLL
CCC	LLL	III	UUU	UUU	TTT	LLL
CCC	LLL	III	UUU	UUU	TTT	LLL
CCC	LLL	III	UUU	UUU	TTT	LLL
CCC	LLL	III	UUU	UUU	TTT	LLL
CCC	LLL	III	UUU	UUU	TTT	LLL
CCC	LLL	III	UUU	UUU	TTT	LLL
CCC	LLL	III	UUU	UUU	TTT	LLL
CCC	LLL	III	UUU	UUU	TTT	LLL
CCC	LLL	III	UUU	UUU	TTT	LLL
CCC	LLL	III	UUU	UUU	TTT	LLL
CCC	LLL	III	UUU	UUU	TTT	LLL
CCC	LLL	III	UUU	UUU	TTT	LLL
CCC	LLL	III	UUU	UUU	TTT	LLL
CCC	LLL	III	UUU	UUU	TTT	LLL
CCC	LLL	III	UUU	UUU	TTT	LLL
CCC	LLL	III	UUU	UUU	TTT	LLL
CCCCCCCCCCCC	LLLLLLLLLLLLLLLL	IIIIIIII	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	TTTT	LLLLLLLLLLLLLLLL
CCCCCCCCCCCC	LLLLLLLLLLLLLLLL	IIIIIIII	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	TTTT	LLLLLLLLLLLLLLLL
CCCCCCCCCCCC	LLLLLLLLLLLLLLLL	IIIIIIII	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	TTTT	LLLLLLLLLLLLLLLL

```
SSSSSSSS EEEEEEEEE TTTTTTTTT AAAAAA CCCCCCC TTTTTTTTT
SSSSSSSS EEEEEEEEE TTTTTTTTT AAAAAA CCCCCCC TTTTTTTTT
SS          EE          TT          AA          AA          CC          TT
SS          EE          TT          AA          AA          CC          TT
SS          EE          TT          AA          AA          CC          TT
SS          EE          TT          AA          AA          CC          TT
SSSSSS    EEEEEEEE    TT          AA          AA          CC          TT
SSSSSS    EEEEEEEE    TT          AA          AA          CC          TT
          SS          EE          AA          AA          CC          TT
          SS          EE          AA          AA          CC          TT
          SS          EE          AA          AA          CC          TT
          SS          EE          AA          AA          CC          TT
SSSSSSSS EEEEEEEEE TTTTTTTTT AA          AA          CCCCCCC TT
SSSSSSSS EEEEEEEEE TTTTTTTTT AA          AA          CCCCCCC TT
          TT          AA          AA          CCCCCCC TT
          TT          AA          AA          CCCCCCC TT
```

```
LL          IIIIII    SSSSSSS
LL          IIIIII    SSSSSSS
L'          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 IIIIII    SSSSSSS
LLLLLLLLLL IIIIII    SSSSSSS
```

```

1 0001 0 MODULE setact (
2 0002 0 IDENT = 'V04-000',
3 0003 0 ADDRESSING_MODE(EXTERNAL=GENERAL,
4 0004 0 NONEXTERNAL=LONG_RELATIVE)
5 0005 0 ) =
6 0006 1 BEGIN
7 0007 1
8 0008 1
9 0009 1 *****
10 0010 1 *
11 0011 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
12 0012 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
13 0013 1 * ALL RIGHTS RESERVED.
14 0014 1 *
15 0015 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
16 0016 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
17 0017 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
18 0018 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
19 0019 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
20 0020 1 * TRANSFERRED.
21 0021 1 *
22 0022 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
23 0023 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
24 0024 1 * CORPORATION.
25 0025 1 *
26 0026 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
27 0027 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
28 0028 1 *
29 0029 1 *
30 0030 1 *****
31 0031 1
32 0032 1
33 0033 1 ++
34 0034 1 FACILITY: Set
35 0035 1
36 0036 1 ABSTRACT:
37 0037 1
38 0038 1 This module contains the action routines for SET FILE, SET DIRECTORY,
39 0039 1 and SET VOLUME.
40 0040 1
41 0041 1 ENVIRONMENT:
42 0042 1
43 0043 1 Vax native, privileged user mode
44 0044 1
45 0045 1 --
46 0046 1
47 0047 1 AUTHOR: Gerry Smith CREATION DATE: 04-Aug-1981
48 0048 1
49 0049 1 MODIFIED BY:
50 0050 1
51 0051 1 V03-005 GAS0047 Gerry Smith 15-Feb-1982
52 0052 1 Only get the file name for SET FILE/ENTER=filename.
53 0053 1 The $PARSE is moved to SETFILE, so that stickiness
54 0054 1 can be applied with the input file.
55 0055 1
56 0056 1 V03-004 GAS0038 Gerry Smith 2-Feb-1982
57 0057 1 Add /GLOBAL_BUFFERS action routine for SET FILE.

```

: 58 0058 1 !  
: 59 0059 1 !  
: 60 0060 1 !  
: 61 0061 1 !  
: 62 0062 1 !  
: 63 0063 1 !  
: 64 0064 1 !  
: 65 0065 1 !  
: 66 0066 1 !  
: 67 0067 1 !  
: 68 0068 1 !  
: 69 0069 1 !\*\*

V03-003 GAS0030 Gerry Smith 1-Jan-1982  
Add /RETENTION action routine, for SET VOLUME.  
V03-002 GAS0026 Gerry Smith 18-Dec-1981  
Use shared message file, and lower fatal messages to  
simple error messages.  
V03-001 GAS0021 Gerry Smith 30-Nov-1981  
Allow zero values for group and member of UIC

SETACT  
V04-000

<sup>1</sup>  
~~16-Sep-1984~~ 01:06:01  
14-Sep-1984 12:08:59

VAX-11 Bliss-32 V4.0-742  
[CLIUTL.SRC]SETACT.B32;1

Page 3  
(2)

```
: 71      0070 1 LIBRARY 'SYSSLIBRARY:LIB':  
: 72      0071 1 LIBRARY 'SYSSLIBRARY:CLIMAC':  
: 73      0072 1 LIBRARY 'SYSSLIBRARY:TPAMAC':  
: 74      0073 1  
: 75      0074 1 STRUCTURE  
: 76      0075 1     BBLOCK [O, P, S, E; N] =  
: 77      0076 1     [N]  
: 78      0077 1     (BBLOCK + O)<P,S,E>:
```

```

80      0078 1 FORWARD ROUTINE
81      0079 1
82      0080 1     acc_act,
83      0081 1     back_act,
84      0082 1     noback_act,
85      0083 1     data_act,
86      0084 1     enter_act,
87      0085 1     erase_act,
88      0086 1     noerase_act,
89      0087 1     exp_act,
90      0088 1     noexp_act,
91      0089 1     ext_act,
92      0090 1     fprot_act,
93      0091 1     gbuf_act,
94      0092 1     journal_act,
95      0093 1     label_act,
96      0094 1     owner_act,
97      0095 1     retent_act,
98      0096 1     test_char,
99      0097 1     user_act,
100     0098 1     vprot_act,
101     0099 1     vrsn_act,
102     0100 1     window_act;
103     0101 1
104     0102 1 EXTERNAL ROUTINE
105     0103 1     calculate_max,
106     0104 1     sys$fa0,
107     0105 1     lib$tparse,
108     0106 1     lib$cv_ttime,
109     0107 1     lib$cv_tdtm,
110     0108 1     lib$cv_tdtb;
111     0109 1
112     0110 1
113     0111 1
114     0112 1
115     0113 1 EXTERNAL
116     0114 1     rename_buf : VECTOR[nam$c_maxrss, BYTE],      ! Name buffer for /ENTER
117     0115 1     file_name : VECTOR[2],                      ! File name descriptor
118     0116 1     file_rlf : BBLOCK[nam$c_bln],                ! Related name block
119     0117 1
120     0118 1     set$l_status,                                ! Status return for SET dispatcher
121     0119 1     set$a_cliwork;                               ! CLI work area in SET dispatcher
122     0120 1
123     0121 1
124     0122 1
125     0123 1
126     0124 1 LITERAL
127     0125 1     true = 1;
128     0126 1     false = 0;
129     0127 1

```

```

! Action routines for:
! /ACCESSED (VOLUME)
! /BACKUP (FILE)
! /NOBACKUP (FILE)
! /DATA_CHECK (VOLUME, FILE)
! /ENTER (FILE)
! /ERASE ON DELETE (FILE)
! /NOERASE ON DELETE (FILE)
! /EXPIRATION_DATE (FILE)
! /NOEXPIRATION_DATE (FILE)
! /EXTENSION (FILE, VOLUME)
! /FILE PROTECTION (VOLUME)
! /GLOBAL BUFFERS (FILE)
! /JOURNAL (FILE)
! /LABEL (VOLUME)
! /OWNER UIC (ALL)
! /RETENTION (VOLUME)
! action routine used by retent_act
! /USER NAME (VOLUME)
! /PROTECTION (VOLUME)
! /VERSION_LIMIT (DIRECTORY, FILE)
! /WINDOWS (VOLUME)

```

```

131 0128 1 |
132 0129 1 | Define the qualifier flag bits used by all SET FILE/DIRECTORY/VOLUME
133 0130 1 |
134 0131 1 | GLOBAL LITERAL
135 P 0132 1 |   SEQUALST
136 P P 0133 1 |     (QUAL_.,1,1,
137 P P 0134 1 |       (access,).
138 P P 0135 1 |       (backup,).
139 P P 0136 1 |       (nobackup,).
140 P P 0137 1 |       (confirm,).
141 P P 0138 1 |       (data,).
142 P P 0139 1 |       (enter,).
143 P P 0140 1 |       (eof,).
144 P P 0141 1 |       (erase,).
145 P P 0142 1 |       (noerase,).
146 P P 0143 1 |       (expi,).
147 P P 0144 1 |       (exte,).
148 P P 0145 1 |       (fprot,).
149 P P 0146 1 |       (gbuf,).
150 P P 0147 1 |       (journal,).
151 P P 0148 1 |       (label,).
152 P P 0149 1 |       (log,).
153 P P 0150 1 |       (nodi,).
154 P P 0151 1 |       (owner,).
155 P P 0152 1 |       (parent,).
156 P P 0153 1 |       (remove,).
157 P P 0154 1 |       (retent,).
158 P P 0155 1 |       (rprot,).
159 P P 0156 1 |       (trunc,).
160 P P 0157 1 |       (username,).
161 P P 0158 1 |       (vprot,).
162 P 0159 1 |       (vrsn,).
163 0160 1 |       (windows,));
164 0161 1 |
165 0162 1 |
166 0163 1 | Define the DATA_CHECK option bits
167 0164 1 |
168 0165 1 | GLOBAL LITERAL
169 P 0166 1 |   SEQUALST
170 P P 0167 1 |     (DATA_.,1,1,
171 P P 0168 1 |       (read,).
172 P P 0169 1 |       (noread,).
173 P 0170 1 |       (write,).
174 0171 1 |       (nowrite,));
175 0172 1 |
176 0173 1 | Define the JOURNAL option bits
177 0174 1 |
178 0175 1 | GLOBAL LITERAL
179 P 0176 1 |   SEQUALST
180 P P 0177 1 |     (JRNL_.,1,1,
181 P P 0178 1 |       (ai,).
182 P P 0179 1 |       (noai,).
183 P P 0180 1 |       (at,).
184 P P 0181 1 |       (noat,).
185 P P 0182 1 |       (bi,).
186 P 0183 1 |       (nobi,).
187 P 0184 1 |       (ru,).

```

```

: ACCESSED (VOLUME)
: BACKUP (FILE)
: NOBACKUP (FILE)
: CONFIRM (ALL)
: DATA_CHECK (FILE, VOLUME)
: ENTER (FILE)
: END OF FILE (FILE)
: ERASE_ON_DELETE (FILE)
: NOERASE_ON_DELETE (FILE)
: EXPIRATION_DATE (FILE)
: EXTENSION (FILE, VOLUME)
: FILE_PROTECTION (VOLUME)
: ! GLOBAL_BUFFERS (FILE)
: JOURNAL (FILE)
: LABEL (VOLUME)
: LOG (ALL)
: NODIRECTORY (FILE)
: OWNER_UIC (ALL)
: OWNER=PARENT (FILE)
: REMOVE (FILE)
: RETENTION (VOLUME)
: RECORD_PROTECTION (FILE, VOLUME)
: TRUNCATE (FILE)
: USERNAME (VOLUME)
: PROTECTION (VOLUME)
: VERSION_LIMIT (FILE, DIRECTORY)
: WINDOWS (VOLUME)

```

SETACT  
V04-000

B 2  
16-Sep-1984 01:06:01  
14-Sep-1984 12:08:59

VAX-11 Bliss-32 V4.0-742  
[CLIUTL.SRC]SETACT.B32;1

Page 6  
(4)

:	188	P 0185	1	(noru,).
:	189	P 0186	1	(rum,).
:	190	0187	1	(norum,));
:	191	0188	1	
:	192	0189	1	



```

: 194      0190  1  !
: 195      0191  1  !: Declare external references
: 196      0192  1  !:
: 197      0193  1  EXTERNAL
: 198      0194  1      setfile$flags : BITVECTOR[32],           ! Qualifier flags
: 199      0195  1      setfile$dflags : BITVECTOR[32],           ! Data check flag word
: 200      0196  1      setfile$jflags : BITVECTOR[32],           ! Journal flag word
: 201      0197  1      acc_value,                               ! LRU value
: 202      0198  1      exp_value : BBLOCK[8],                   ! Expiration date
: 203      0199  1      exte_value,                               ! Extension quantity
: 204      0200  1      fprof_value,                              ! File protection value
: 205      0201  1      gbuf_value,                               ! Global buffer count
: 206      0202  1      label_value : VECTOR[2],                 ! Volume label descriptor
: 207      0203  1      uic_value,                                ! Owner uic
: 208      0204  1      group,                                   ! Group number
: 209      0205  1      member,                                  ! Member number
: 210      0206  1      user_value : VECTOR[2],                  ! Username string descriptor
: 211      0207  1      retmin_value : VECTOR[2],                ! Minimum retention time quadword
: 212      0208  1      retmax_value : VECTOR[2],                ! Maximum retention time quadword
: 213      0209  1      vprot_value,                             ! Volume protection value
: 214      0210  1      vrsn_value,                              ! Version limit
: 215      0211  1      window_value;                            ! Window length
: 216      0212  1
: 217      0213  1
: 218      0214  1  !
: 219      0215  1  !: Declare the error messages
: 220      0216  1  !:
: 221      0217  1  EXTERNAL LITERAL
: 222      0218  1      set$_facility,                            ! SET facility code
: 223      0219  1      set$_operreq,                            ! OPER privilege required
: 224      0220  1      set$_writeerr;                          ! Error accessing file

```



```
236 0230 1 !  
237 0231 1 ! TPARSE table for /DATA_CHECK options  
238 0232 1 !  
239 0233 1 $INIT_STATE (dc_state,dc_keys);  
240 0234 1  
241 P 0235 1 $STATE (optstart,  
242 P 0236 1 (tpa$_eos, tpa$_exit, , data_write, setfile$dflags), ! default is WRITE  
243 0237 1 (tpa$_lambda, getoption)); ! fall thru to options  
244 0238 1  
245 P 0239 1 $STATE (getoption,  
246 P 0240 1 ('READ', , , 1^data_read, setfile$dflags), ! READ present  
247 P 0241 1 ('WRITE', , , 1^data_write, setfile$dflags), ! WRITE present  
248 P 0242 1 ('NOREAD', , , 1^data_noread, setfile$dflags), ! NOREAD present  
249 0243 1 ('NOWRITE', , , 1^data_nowrite, setfile$dflags)); ! NOWRITE present  
250 0244 1  
251 P 0245 1 $STATE (  
252 P 0246 1 (tpa$_eos, tpa$_exit), ! either end of line  
253 0247 1 ('', , , getoption)); ! or get rid of the comma  
254 0248 1
```

```

: 256      0249 1  |
: 257      0250 1  | TPARSE table for /OWNER_UIC option
: 258      0251 1  |
: 259      0252 1  | $INIT_STATE (owner_state,owner_keys);
: 260      0253 1  |
: 261      P 0254 1  | $STATE (ownerstart,
: 262      P 0255 1  | ('PARENT',tpa$_exit,,1^qual_parent,setfile$flags), ! Check for PARENT
: 263      P 0256 1  | ('['), ! Look for square bracket
: 264      0257 1  | ('<')); ! Or a squiggle
: 265      0258 1  |
: 266      P 0259 1  | $STATE (,
: 267      0260 1  | (tpa$_octal,,,,group)); ! Get group number
: 268      0261 1  |
: 269      P 0262 1  | $STATE (,')'); ! Get rid of the comma
: 270      0263 1  |
: 271      0264 1  |
: 272      P 0265 1  | $STATE (,
: 273      0266 1  | (tpa$_octal,,,,member)); ! Get member number
: 274      0267 1  |
: 275      P 0268 1  | $STATE (,
: 276      P 0269 1  | (]'); ! Get end bracket
: 277      0270 1  | ('>'));
: 278      0271 1  |
: 279      P 0272 1  | $STATE (,
: 280      0273 1  | (tpa$_eos,tpa$_exit)); ! Clean-up

```

```

: 282      0274 1  |
: 283      0275 1  | TPARSE table for /JOURNAL option
: 284      0276 1  |
: 285      0277 1  | $INIT_STATE (journal_state, journal_keys);
: 286      0278 1  |
: 287      P 0279 1  | $STATE (getjopts
: 288      P P 0280 1  | ('AI'...1^jrn[_ai,setfile$]flags), | AI journaling
: 289      P P 0281 1  | ('AT'...1^jrn[_at,setfile$]flags), | AT journaling
: 290      P P 0282 1  | ('BI'...1^jrn[_bi,setfile$]flags), | BI journaling
: 291      P P 0283 1  | ('NOAI'...1^jrn[_noai,setfile$]flags), | No AI journaling
: 292      P P 0284 1  | ('NOAT'...1^jrn[_noat,setfile$]flags), | No AT journaling
: 293      P P 0285 1  | ('NOBI'...1^jrn[_nobi,setfile$]flags), | No BI journaling
: 294      P P 0286 1  | ('NORU'...1^jrn[_noru,setfile$]flags), | No RU journaling
: 295      P P 0287 1  | ('NORUM'...1^jrn[_norum,setfile$]flags), | No RUM journaling
: 296      P 0288 1  | ('RU'...1^jrn[_ru,setfile$]flags), | RU journaling
: 297      0289 1  | ('RUM'...1^jrn[_rum,setfile$]flags)); | RUM journaling
: 298      0290 1  |
: 299      P 0291 1  | $STATE (
: 300      P 0292 1  | (tpa$ eos, tpa$ exit), | Either the end
: 301      0293 1  | ('.', getjopts)); | Or more to come

```

```

303      0294 1 |
304      0295 1 | TPARSE table for protection
305      0296 1 |
306      0297 1 |
307      0298 1 $INIT_STATE (pro_state, pro_keys);
308      0299 1
309      P 0300 1 $STATE (NEXTPRO
310      P P 0301 1 ('SYSTEM', SYPR,, %X'00F0000', fprot_value),
311      P P 0302 1 ('OWNER', OWPR,, %X'00F00000', fprot_value),
312      P P 0303 1 ('GROUP', GRPR,, %X'0F000000', fprot_value),
313      P 0304 1 ('WORLD', WOPR,, %X'F0000000', fprot_value)
314      0305 1 );
315      0306 1
316      P 0307 1 $STATE (SYPR,
317      P P 0308 1 (':'),
318      P P 0309 1 ('='),
319      P 0310 1 (TPAS_LAMBDA, ENDPRO)
320      0311 1 );
321      0312 1
322      P 0313 1 $STATE (SYPRO,
323      P P 0314 1 ('R', SYPRO,, %X'0001', fprot_value),
324      P P 0315 1 ('W', SYPRO,, %X'0002', fprot_value),
325      P P 0316 1 ('E', SYPRO,, %X'0004', fprot_value),
326      P P 0317 1 ('P', SYPRO,, %X'0004', fprot_value),
327      P P 0318 1 ('D', SYPRO,, %X'0008', fprot_value),
328      P 0319 1 ('L', SYPRO,, %X'0008', fprot_value),
329      P 0320 1 (TPAS_LAMBDA, ENDPRO)
330      0321 1 );
331      0322 1
332      P 0323 1 $STATE (OWPR,
333      P P 0324 1 (':'),
334      P P 0325 1 ('='),
335      P 0326 1 (TPAS_LAMBDA, ENDPRO)
336      0327 1 );
337      0328 1
338      P 0329 1 $STATE (OWPRO,
339      P P 0330 1 ('R', OWPRO,, %X'0010', fprot_value),
340      P P 0331 1 ('W', OWPRO,, %X'0020', fprot_value),
341      P P 0332 1 ('E', OWPRO,, %X'0040', fprot_value),
342      P P 0333 1 ('P', OWPRO,, %X'0040', fprot_value),
343      P P 0334 1 ('D', OWPRO,, %X'0080', fprot_value),
344      P P 0335 1 ('L', OWPRO,, %X'0080', fprot_value),
345      P 0336 1 (TPAS_LAMBDA, ENDPRO)
346      0337 1 );
347      0338 1
348      P 0339 1 $STATE (GRPR,
349      P P 0340 1 (':'),
350      P P 0341 1 ('='),
351      P 0342 1 (TPAS_LAMBDA, ENDPRO)
352      0343 1 );
353      0344 1
354      P 0345 1 $STATE (GRPRO,
355      P 0346 1 ('R', GRPRO,, %X'0100', fprot_value),
356      P 0347 1 ('W', GRPRO,, %X'0200', fprot_value),
357      P 0348 1 ('E', GRPRO,, %X'0400', fprot_value),
358      P 0349 1 ('P', GRPRO,, %X'0400', fprot_value),
359      P 0350 1 ('D', GRPRO,, %X'0800', fprot_value).

```

```
360 P 0351 1 ('L' GRPRO,, ZX'0800', fprot_value),
361 P 0352 1 (TPAS_LAMBDA, ENDPRO)
362 P 0353 1 );
363 P 0354 1
364 P 0355 1 $STATE (WOPR,
365 P 0356 1 (':'),
366 P 0357 1 ('='),
367 P 0358 1 (TPAS_LAMBDA, ENDPRO)
368 P 0359 1 );
369 P 0360 1
370 P 0361 1 $STATE (WOPRO,
371 P 0362 1 ('R' WOPRO,, ZX'1000', fprot_value),
372 P 0363 1 ('W' WOPRO,, ZX'2000', fprot_value),
373 P 0364 1 ('E' WOPRO,, ZX'4000', fprot_value),
374 P 0365 1 ('P' WOPRO,, ZX'4000', fprot_value),
375 P 0366 1 ('D' WOPRO,, ZX'8000', fprot_value),
376 P 0367 1 ('L' WOPRO,, ZX'8000', fprot_value),
377 P 0368 1 (TPAS_LAMBDA, ENDPRO)
378 P 0369 1 );
379 P 0370 1
380 P 0371 1 $STATE (ENDPRO,
381 P 0372 1 (' ' NÉXTPRO)
382 P 0373 1 (TPAS_EOS, TPAS_EXIT)
383 P 0374 1 );
```

```

: 385      0375 1  |
: 386      0376 1  | | Tparse table for /RETENTION option
: 387      0377 1  |
: 388      0378 1  | $INIT_STATE (ret_state, ret_keys)
: 389      0379 1  |
: 390      P 0380 1  | $STATE (retstart,
: 391          ((get_delta),...,retmin_value));           ! Get the first delta string
: 392          0381 1  |
: 393      P 0382 1  | $STATE (
: 394          (f,'),                                       ! If a comma, get next string
: 395          (tpa$_eos, tpa$_exit));                       ! Else exit
: 396          0383 1  |
: 397      P 0384 1  | $STATE (
: 398          ((get_delta),...,retmax_value));           ! Get the next delta string
: 399          0385 1  |
: 400      P 0386 1  | $STATE (
: 401          (tpa$_eos, tpa$_exit));                       ! And exit
: 402          0387 1  |
: 403      P 0388 1  | $STATE (get_delta,
: 404          (tpa$_any, get_delta, test_char),             ! Get next character in delta string
: 405          0389 1  | (tpa$_lambda, tpa$_exit));

```



```

407 0396 1 GLOBAL ROUTINE acc_act (option_block, callback) =
408 0397 1 |++
409 0398 1 |
410 0399 1 | This is the action routine for the /ACCESSED qualifier. It first checks to
411 0400 1 | make sure that the process has OPER privilege. If so, then the ACCESS value
412 0401 1 | is obtained and bounds checking is performed on it.
413 0402 1 |
414 0403 1 |--
415 0404 2 BEGIN
416 0405 2
417 0406 2 OWN privs : BBLOCK[8];           ! Place to store the process privileges
418 0407 2
419 0408 2 LOCAL
420 0409 2     status,                       ! Status return
421 0410 2     desc : BBLOCK[dsc$s_bln];    ! General descriptor
422 0411 2
423 0412 2 MAP option_block : REF BBLOCK; ! Define the CLI block
424 0413 2
425 0414 2 |
426 0415 2 | Call $SETPRV to get the current privileges of the process. If the process
427 0416 2 | does not have OPER, then signal an error and stop.
428 0417 2
429 P 0418 2 IF NOT (status = $SETPRV(ENBFLG = 1,           ! Enable
430 PP 0419 2     PRVADR = 0,                               ! No new privileges
431 P 0420 2     PRMFLG = 1,                               ! Get current privileges
432 0421 2     PRVPRV = privs))
433 0422 2 THEN SIGNAL_STOP(.status);
434 0423 2
435 0424 2 IF NOT .privs[prv$v_oper] THEN SIGNAL_STOP(set$_operreq);
436 0425 2
437 0426 2 |
438 0427 2 | The process has the correct privilege, so go ahead and get the value
439 0428 2 |
440 0429 2
441 0430 2 acc_value = 3;                       ! Set up the default
442 0431 2
443 0432 2 |
444 0433 2 | If a value was specified, use it; otherwise, use the default.
445 0434 2 |
446 0435 2 IF .option_block[cli$w_qdvalsiz] EQL 0
447 0436 2 THEN RETURN true;
448 0437 2
449 0438 2 |
450 0439 2 | Convert the value
451 0440 2 |
452 0441 2 IF NOT (status = LIB$CVT_DTB(.option_block[cli$w_qdvalsiz],
453 0442 2     .option_block[cli$a_qdvaladr],
454 0443 2     acc_value))
455 0444 2 THEN SIGNAL_STOP(set$_facility*16 + shr$_syntax + sts$_error, ! Signal a syntax error
456 0445 2     1,
457 0446 2     option_block[cli$q_qvaldesc],
458 0447 2     .status)
459 0448 2 ELSE
460 0449 2     BEGIN
461 0450 2     IF NOT (.acc_value GEQ 0           ! Check that value is in range
462 0451 2     AND
463 0452 2     .acc_value LEQ 255)

```

```
: 464      0453 3      THEN SIGNAL_STOP(set$_facility^16 + shr$_syntax + sts$_k_error,      ! If not, exit with an error.
: 465      0454 3
: 466      0455 3      option_block[cli$_q_dvaldesc],
: 467      0456 3      set$_facility^16 + shr$_valerr + sts$_k_error);
: 468      0457 2      END;
: 469      0458 2      RETURN true;
: 470      0459 1      END;
```

		.TITLE	SETACT				
		.IDENT	\V04-000\				
		.PSECT	_LIB\$KEY1\$,NOWRT,	SHR,	PIC,1		
		00000	:TPASKEYSTO				
		U.9:	.BLKB	0			
44	41	45	52	00000	:TPASKEYST		
		FF	00004	.ASCII	\READ\		
			00005	.BYTE	-1		
			U.15:	.BLKB	0		
45	54	49	52	57	00005		
		FF	0000A	.ASCII	\WRITE\		
			0000B	.BYTE	-1		
			U.21:	.BLKB	0		
44	41	45	52	4F	4E	0000B	
		FF	00011	.ASCII	\NOREAD\		
			00012	.BYTE	-1		
			U.27:	.BLKB	0		
45	54	49	52	57	4F	4E	00012
		FF	00019	.ASCII	\NOWRITE\		
			0001A	.BYTE	-1		
			U.33:	.BYTE	-1		
			0001B	.BLKB	0		
54	4E	45	52	41	50	0001B	
		FF	00021	.ASCII	\PARENT\		
			00022	.BYTE	-1		
			U.48:	.BYTE	-1		
			00023	.BLKB	0		
			U.59:	.BLKB	0		
49	41	00023	:TPASKEYST				
		FF	00025	.ASCII	\AI\		
			00026	.BYTE	-1		
			U.65:	.BLKB	0		
54	41	00026	:TPASKEYST				
		FF	00028	.ASCII	\AT\		
			00029	.BYTE	-1		
			U.71:	.BLKB	0		
49	42	00029	:TPASKEYST				
			U.73:	.ASCII	\BI\		

				FF	0002B		:TPASKEYSTO	BYTE	-1	:
					0002C		U.77: .BLKB	0		:
49	41	4F	4E	0002C		U.79: .ASCII	\NOAI\			:
				FF	00030		:TPASKEYSTO	BYTE	-1	:
					00031		U.83: .BLKB	0		:
54	41	4F	4E	00031		U.85: .ASCII	\NCAT\			:
				FF	00035		:TPASKEYSTO	BYTE	-1	:
					00036		U.89: .BLKB	0		:
49	42	4F	4E	00036		U.91: .ASCII	\NOBI\			:
				FF	0003A		:TPASKEYSTO	BYTE	-1	:
					0003B		U.95: .BLKB	0		:
55	52	4F	4E	0003B		U.97: .ASCII	\NORU\			:
				FF	0003F		:TPASKEYSTO	BYTE	-1	:
					00040		U.101: .BLKB	0		:
4D	55	52	4F	4E	00040		U.103: .ASCII	\NORUM\		:
				FF	00045		:TPASKEYSTO	BYTE	-1	:
					00046		U.107: .BLKB	0		:
			55	52	00046		U.109: .ASCII	\RU\		:
				FF	00048		:TPASKEYSTO	BYTE	-1	:
					00049		U.113: .BLKB	0		:
		4D	55	52	00049		U.115: .ASCII	\RUM\		:
				FF	0004C		:TPASKEYFILL	BYTE	-1	:
				FF	0004D		U.119: .BLKB	0		:
					0004E		U.125: .ASCII	\SYSTEM\		:
4D	45	54	53	59	53	0004E	U.127: .ASCII	\SYSTEM\	-1	:
				FF	00054		:TPASKEYSTO	BYTE	-1	:
					00055		U.133: .BLKB	0		:
52	45	4E	57	4F	00055		U.135: .ASCII	\OWNER\		:
				FF	0005A		:TPASKEYSTO	BYTE	-1	:
					0005B		U.141: .BLKB	0		:
50	55	4F	52	47	0005B		U.143: .ASCII	\GROUP\		:
				FF	00060		:TPASKEYSTO	BYTE	-1	:
					00061		U.149: .BLKB	0		:
44	4C	52	4F	57	00061		U.151: .ASCII	\WORLD\		:

```
FF 00066 .BYTE -1 ;
FF 00067 :TPASKEYFILL .BYTE -1 ;
                                U.157: .BYTE -1 ;
                                .PSECT _LIB$STATES,NOWRT, SHR, PIC,1
00000 DC_STATE::
                                .BLKB 0
00000 OPTSTART:
                                .BLKB 0
71F7 00000 :TPASTYPE
                                .WORD 29175 ;
00000000* 00002 :TPASADDR
                                U.2: .WORD 29175 ;
                                U.3: .LONG <<SETFILES$DFLAGS-U.3>-4> ;
00000003 00006 :TPASMASK
                                U.4: .LONG 3 ;
FFFF 0000A :TPASTARGET
                                U.5: .WORD -1 ;
15F6 0000C :TPASTYPE
                                U.6: .WORD 5622 ;
0000* 0000E :TPASTARGET
                                U.8: .WORD <<U.7-U.8>-2> ;
                                00010 :GETOPTION
                                U.7: .BLKB 0
6100 00010 :TPASTYPE
                                U.12: .WORD 24832 ;
00000000* 00012 :TPASADDR
                                U.13: .LONG <<SETFILES$DFLAGS-U.13>-4> ;
00000002 00016 :TPASMASK
                                U.14: .LONG 2 ;
6101 0001A :TPASTYPE
                                U.18: .WORD 24833 ;
00000000* 0001C :TPASADDR
                                U.19: .LONG <<SETFILES$DFLAGS-U.19>-4> ;
00000008 00020 :TPASMASK
                                U.20: .LONG 8 ;
6102 00024 :TPASTYPE
                                U.24: .WORD 24834 ;
00000000* 00026 :TPASADDR
                                U.25: .LONG <<SETFILES$DFLAGS-U.25>-4> ;
00000004 0002A :TPASMASK
                                U.26: .LONG 4 ;
6503 0002E :TPASTYPE
                                U.30: .WORD 25859 ;
00000000* 00030 :TPASADDR
                                U.31: .LONG <<SETFILES$DFLAGS-U.31>-4> ;
00000010 00034 :TPASMASK
                                U.32: .LONG 16 ;
11F7 00038 :TPASTYPE
                                U.34: .WORD 4599 ;
FFFF 0003A :TPASTARGET
                                U.35: .WORD -1 ;
142C 0003C :TPASTYPE
                                U.36: .WORD 5164 ;
0000* 0003E :TPASTARGET
                                U.37: .WORD <<U.7-U.37>-2> ;
00040 OWNER_STATE::
```

			.BLKB	0	
	00040	OWNERSTART:			
			.BLKB	0	
7100	00040	:TPASTYPE			
		U.42:	.WORD	28928	:
00000000*	00042	:TPASADDR			:
		U.43:	.LONG	<<SETFILES\$FLAGS-U.43>-4>	:
00080000	00046	:TPASMASK			:
		U.44:	.LONG	524288	:
FFFF	0004A	:TPASTARGET			:
		U.45:	.WORD	-1	:
005B	0004C	:TPASTYPE			:
		U.46:	.WORD	91	:
043C	0004E	:TPASTYPE			:
		U.47:	.WORD	1084	:
45F4	00050	:TPASTYPE			:
		U.49:	.WORD	17908	:
00000000*	00052	:TPASADDR			:
		U.50:	.LONG	<<GROUP-U.50>-4>	:
042C	00056	:TPASTYPE			:
		U.51:	.WORD	1068	:
45F4	00058	:TPASTYPE			:
		U.52:	.WORD	17908	:
00000000*	0005A	:TPASADDR			:
		U.53:	.LONG	<<MEMBER-U.53>-4>	:
005D	0005E	:TPASTYPE			:
		U.54:	.WORD	93	:
043E	00060	:TPASTYPE			:
		U.55:	.WORD	1086	:
15F7	00062	:TPASTYPE			:
		U.56:	.WORD	5623	:
FFFF	00064	:TPASTARGET			:
		U.57:	.WORD	-1	:
	00066		.BLKB	2	:
	00068	JOURNAL_STATE:			:
			.BLKB	0	:
	00068	GETJOPTS:			:
			.BLKB	0	:
6100	00068	:TPASTYPE			:
		U.62:	.WORD	24832	:
00000000*	0006A	:TPASADDR			:
		U.63:	.LONG	<<SETFILES\$JFLAGS-U.63>-4>	:
00000002	0006E	:TPASMASK			:
		U.64:	.LONG	2	:
6101	00072	:TPASTYPE			:
		U.68:	.WORD	24833	:
00000000*	00074	:TPASADDR			:
		U.69:	.LONG	<<SETFILES\$JFLAGS-U.69>-4>	:
00000008	00078	:TPASMASK			:
		U.70:	.LONG	8	:
6102	0007C	:TPASTYPE			:
		U.74:	.WORD	24834	:
00000000*	0007E	:TPASADDR			:
		U.75:	.LONG	<<SETFILES\$JFLAGS-U.75>-4>	:
00000020	00082	:TPASMASK			:
		U.76:	.LONG	32	:
6103	00086	:TPASTYPE			:

00000000*	00088	U.80:	WORD	24835	:
		:TPASADDR			:
00000004	0008C	U.81:	LONG	<<SETFILESJFLAGS-U.81>-4>	:
		:TPASMASK			:
6104	00090	U.82:	LONG	4	:
		:TPASTYPE			:
00000000*	00092	U.86:	WORD	24836	:
		:TPASADDR			:
00000010	00096	U.87:	LONG	<<SETFILESJFLAGS-U.87>-4>	:
		:TPASMASK			:
6105	0009A	U.88:	LONG	16	:
		:TPASTYPE			:
00000000*	0009C	U.92:	WORD	24837	:
		:TPASADDR			:
00000040	000A0	U.93:	LONG	<<SETFILESJFLAGS-U.93>-4>	:
		:TPASMASK			:
6106	000A4	U.94:	LONG	64	:
		:TPASTYPE			:
00000000*	000A6	U.98:	WORD	24838	:
		:TPASADDR			:
00000100	000AA	U.99:	LONG	<<SETFILESJFLAGS-U.99>-4>	:
		:TPASMASK			:
6107	000AE	U.100:	LONG	256	:
		:TPASTYPE			:
00000000*	000B0	U.104:	WORD	24839	:
		:TPASADDR			:
00000400	000B4	U.105:	LONG	<<SETFILESJFLAGS-U.105>-4>	:
		:TPASMASK			:
6108	000B8	U.106:	LONG	1024	:
		:TPASTYPE			:
00000000*	000BA	U.110:	WORD	24840	:
		:TPASADDR			:
00000080	000BE	U.111:	LONG	<<SETFILESJFLAGS-U.111>-4>	:
		:TPASMASK			:
6509	000C2	U.112:	LONG	128	:
		:TPASTYPE			:
00000000*	000C4	U.116:	WORD	25865	:
		:TPASADDR			:
00000200	000C8	U.117:	LONG	<<SETFILESJFLAGS-U.117>-4>	:
		:TPASMASK			:
11F7	000CC	U.118:	LONG	512	:
		:TPASTYPE			:
FFFF	000CE	U.120:	WORD	4599	:
		:TPASTARGET			:
142C	000D0	U.121:	WORD	-1	:
		:TPASTYPE			:
0000*	000D2	U.122:	WORD	5164	:
		:TPASTARGET			:
	000D4	U.123:	WORD	<<GETJOPTS-U.123>-2>	:
		PRO_STATE:			:
		:BLKB		0	:
	000D4	NEXTPRO:	BLKB	0	:
7100	000D4	:TPASTYPE			:
		U.128:	WORD	28928	:
00000000*	000D6	:TPASADDR			:
		U.129:	LONG	<<FPROT_VALUE-U.129>-4>	:
000F0000	000DA	:TPASMASK			:

0000*	000DE	U.130: .LONG	983040	:
		:TPASTARGET		
7101	000E0	U.132: .WORD	<<U.131-U.132>-2>	:
		:TPASTYPE		
00000000*	000E2	U.136: .WORD	28929	:
		:TPASADDR		
00F00000	000E6	U.137: .LONG	<<FPROT_VALUE-U.137>-4>	:
		:TPASMASK		
0000*	000EA	U.138: .LONG	15728640	:
		:TPASTARGET		
7102	000EC	U.140: .WORD	<<U.139-U.140>-2>	:
		:TPASTYPE		
00000000*	000EE	U.144: .WORD	28930	:
		:TPASADDR		
0F000000	000F2	U.145: .LONG	<<FPROT_VALUE-U.145>-4>	:
		:TPASMASK		
0000*	000F6	U.146: .LONG	251658240	:
		:TPASTARGET		
7503	000F8	U.148: .WORD	<<U.147-U.148>-2>	:
		:TPASTYPE		
00000000*	000FA	U.152: .WORD	29955	:
		:TPASADDR		
F0000000	000FE	U.153: .LONG	<<FPROT_VALUE-U.153>-4>	:
		:TPASMASK		
0000*	00102	U.154: .LONG	-268435456	:
		:TPASTARGET		
	00104	U.156: .WORD	<<U.155-U.156>-2>	:
		:SYPR		
003A	00104	U.131: .BLKB	0	:
		:TPASTYPE		
003D	00106	U.158: .WORD	58	:
		:TPASTYPE		
15F6	00108	U.159: .WORD	61	:
		:TPASTYPE		
0000*	0010A	U.160: .WORD	5622	:
		:TPASTARGET		
	0010C	U.162: .WORD	<<U.161-U.162>-2>	:
		:SYPRO: .BLKB	0	
7052	0010C	U.163: .WORD	28754	:
		:TPASTYPE		
00000000*	0010E	U.164: .LONG	<<FPROT_VALUE-U.164>-4>	:
		:TPASADDR		
00000001	00112	U.165: .LONG	1	:
		:TPASMASK		
0000*	00116	U.166: .WORD	<<SYPRO-U.166>-2>	:
		:TPASTARGET		
7057	00118	U.167: .WORD	28759	:
		:TPASTYPE		
00000000*	0011A	U.168: .LONG	<<FPROT_VALUE-U.168>-4>	:
		:TPASADDR		
00000002	0011E	U.169: .LONG	2	:
		:TPASMASK		
0000*	00122	U.170: .WORD	<<SYPRO-U.170>-2>	:
		:TPASTARGET		
7045	00124	U.171: .WORD	28741	:
		:TPASTYPE		
00000000*	00126	U.171: .WORD	28741	:
		:TPASADDR		

00000004	0012A	U.172: .LONG	<<FPROT_VALUE-U.172>-4>	:
		:TPASMASK		
		U.173: .LONG	4	:
0000*	0012E	:TPASTARGET		:
		U.174: .WORD	<<SYPRO-U.174>-2>	:
7050	00130	:TPASTYPE		:
		U.175: .WORD	28752	:
00000000*	00132	:TPASADDR		:
		U.176: .LONG	<<FPROT_VALUE-U.176>-4>	:
00000004	00136	:TPASMASK		:
		U.177: .LONG	4	:
0000*	0013A	:TPASTARGET		:
		U.178: .WORD	<<SYPRO-U.178>-2>	:
7044	0013C	:TPASTYPE		:
		U.179: .WORD	28740	:
00000000*	0013E	:TPASADDR		:
		U.180: .LONG	<<FPROT_VALUE-U.180>-4>	:
00000008	00142	:TPASMASK		:
		U.181: .LONG	8	:
0000*	00146	:TPASTARGET		:
		U.182: .WORD	<<SYPRO-U.182>-2>	:
704C	00148	:TPASTYPE		:
		U.183: .WORD	28748	:
00000000*	0014A	:TPASADDR		:
		U.184: .LONG	<<FPROT_VALUE-U.184>-4>	:
00000008	0014E	:TPASMASK		:
		U.185: .LONG	8	:
0000*	00152	:TPASTARGET		:
		U.186: .WORD	<<SYPRO-U.186>-2>	:
15F6	00154	:TPASTYPE		:
		U.187: .WORD	5622	:
0000*	00156	:TPASTARGET		:
		U.188: .WORD	<<U.161-U.188>-2>	:
	00158	:OWPR		:
		U.139: .BLKS	0	:
003A	00158	:TPASTYPE		:
		U.189: .WORD	58	:
003D	0015A	:TPASTYPE		:
		U.190: .WORD	61	:
15F6	0015C	:TPASTYPE		:
		U.191: .WORD	5622	:
0000*	0015E	:TPASTARGET		:
		U.192: .WORD	<<U.161-U.192>-2>	:
	00160	:OWPRO: .BLKB	0	:
7052	00160	:TPASTYPE		:
		U.193: .WORD	28754	:
00000000*	00162	:TPASADDR		:
		U.194: .LONG	<<FPROT_VALUE-U.194>-4>	:
00000010	00166	:TPASMASK		:
		U.195: .LONG	16	:
0000*	0016A	:TPASTARGET		:
		U.196: .WORD	<<OWPRO-U.196>-2>	:
7057	0016C	:TPASTYPE		:
		U.197: .WORD	28759	:
00000000*	0016E	:TPASADDR		:
		U.198: .LONG	<<FPROT_VALUE-U.198>-4>	:
00000020	00172	:TPASMASK		:



0000*	00176	U.199: .LONG	32	:
		:TPASTARGET		:
7045	00178	U.200: .WORD	<<OWPRO-U.200>-2>	:
		:TPASTYPE		:
00000000*	0017A	U.201: .WORD	28741	:
		:TPASADDR		:
00000040	0017E	U.202: .LONG	<<FPROT_VALUE-U.202>-4>	:
		:TPASMASK		:
0000*	00182	U.203: .LONG	64	:
		:TPASTARGET		:
7050	00184	U.204: .WORD	<<OWPRO-U.204>-2>	:
		:TPASTYPE		:
00000000*	00186	U.205: .WORD	28752	:
		:TPASADDR		:
00000040	0018A	U.206: .LONG	<<FPROT_VALUE-U.206>-4>	:
		:TPASMASK		:
0000*	0018E	U.207: .LONG	64	:
		:TPASTARGET		:
7044	00190	U.208: .WORD	<<OWPRO-U.208>-2>	:
		:TPASTYPE		:
00000000*	00192	U.209: .WORD	28740	:
		:TPASADDR		:
00000080	00196	U.210: .LONG	<<FPROT_VALUE-U.210>-4>	:
		:TPASMASK		:
0000*	0019A	U.211: .LONG	128	:
		:TPASTARGET		:
704C	0019C	U.212: .WORD	<<OWPRO-U.212>-2>	:
		:TPASTYPE		:
00000000*	0019E	U.213: .WORD	28748	:
		:TPASADDR		:
00000080	001A2	U.214: .LONG	<<FPROT_VALUE-U.214>-4>	:
		:TPASMASK		:
0000*	001A6	U.215: .LONG	128	:
		:TPASTARGET		:
15F6	001A8	U.216: .WORD	<<OWPRO-U.216>-2>	:
		:TPASTYPE		:
0000*	001AA	U.217: .WORD	5622	:
		:TPASTARGET		:
	001AC	U.218: .WORD	<<U.161-U.218>-2>	:
		:GRPR		:
003A	001AC	U.147: .BLKB	0	:
		:TPASTYPE		:
003D	001AE	U.219: .WORD	58	:
		:TPASTYPE		:
15F6	001B0	U.220: .WORD	61	:
		:TPASTYPE		:
0000*	001B2	U.221: .WORD	5622	:
		:TPASTARGET		:
	001B4	U.222: .WORD	<<U.161-U.222>-2>	:
		:GRPRO: .BLKB	0	:
7052	001B4	:TPASTYPE		:
00000000*	001B6	U.223: .WORD	28754	:
		:TPASADDR		:
00000100	001BA	U.224: .LONG	<<FPROT_VALUE-U.224>-4>	:
		:TPASMASK		:
0000*	001BE	U.225: .LONG	256	:
		:TPASTARGET		:

7057	001C0	U.226: .WORD	<<GRPRO-U.226>-2>	:
		:TPASTYPE		:
00000000*	001C2	U.227: .WORD	28759	:
		:TPASADDR		:
00000200	001C6	U.228: .LONG	<<FPROT_VALUE-U.228>-4>	:
		:TPASMASK		:
0000*	001CA	U.229: .LONG	512	:
		:TPASTARGET		:
7045	001CC	U.230: .WORD	<<GRPRO-U.230>-2>	:
		:TPASTYPE		:
00000000*	001CE	U.231: .WORD	28741	:
		:TPASADDR		:
00000400	001D2	U.232: .LONG	<<FPROT_VALUE-U.232>-4>	:
		:TPASMASK		:
0000*	001D6	U.233: .LONG	1024	:
		:TPASTARGET		:
7050	001D8	U.234: .WORD	<<GRPRO-U.234>-2>	:
		:TPASTYPE		:
00000000*	001DA	U.235: .WORD	28752	:
		:TPASADDR		:
00000400	001DE	U.236: .LONG	<<FPROT_VALUE-U.236>-4>	:
		:TPASMASK		:
0000*	001E2	U.237: .LONG	1024	:
		:TPASTARGET		:
7044	001E4	U.238: .WORD	<<GRPRO-U.238>-2>	:
		:TPASTYPE		:
00000000*	001E6	U.239: .WORD	28740	:
		:TPASADDR		:
00000800	001EA	U.240: .LONG	<<FPROT_VALUE-U.240>-4>	:
		:TPASMASK		:
0000*	001EE	U.241: .LONG	2048	:
		:TPASTARGET		:
704C	001F0	U.242: .WORD	<<GRPRO-U.242>-2>	:
		:TPASTYPE		:
00000000*	001F2	U.243: .WORD	28748	:
		:TPASADDR		:
00000800	001F6	U.244: .LONG	<<FPROT_VALUE-U.244>-4>	:
		:TPASMASK		:
0000*	001FA	U.245: .LONG	2048	:
		:TPASTARGET		:
15F6	001FC	U.246: .WORD	<<GRPRO-U.246>-2>	:
		:TPASTYPE		:
0000*	001FE	U.247: .WORD	5622	:
		:TPASTARGET		:
	00200	U.248: .WORD	<<U.161-U.248>-2>	:
		:WOPR		:
003A	00200	U.155: .BLKB	0	:
		:TPASTYPE		:
003D	00202	U.249: .WORD	58	:
		:TPASTYPE		:
15F6	00204	U.250: .WORD	61	:
		:TPASTYPE		:
0000*	00206	U.251: .WORD	5622	:
		:TPASTARGET		:
	00208	U.252: .WORD	<<U.161-U.252>-2>	:
		:WOPRO: .BLKB	0	:
7052	00208	:TPASTYPE		:

00000000*	0020A	U.253: .WORD	28754	:
		:TPASADDR		:
00001000	0020E	U.254: .LONG	<<FPROT_VALUE-U.254>-4>	:
		:TPASMASK		:
0000*	00212	U.255: .LONG	4096	:
		:TPASTARGET		:
7057	00214	U.256: .WORD	<<WOPRO-U.256>-2>	:
		:TPASTYPE		:
00000000*	00216	U.257: .WORD	28759	:
		:TPASADDR		:
00002000	0021A	U.258: .LONG	<<FPROT_VALUE-U.258>-4>	:
		:TPASMASK		:
0000*	0021E	U.259: .LONG	8192	:
		:TPASTARGET		:
7045	00220	U.260: .WORD	<<WOPRO-U.260>-2>	:
		:TPASTYPE		:
00000000*	00222	U.261: .WORD	28741	:
		:TPASADDR		:
00004000	00226	U.262: .LONG	<<FPROT_VALUE-U.262>-4>	:
		:TPASMASK		:
0000*	0022A	U.263: .LONG	16384	:
		:TPASTARGET		:
7050	0022C	U.264: .WORD	<<WOPRO-U.264>-2>	:
		:TPASTYPE		:
00000000*	0022E	U.265: .WORD	28752	:
		:TPASADDR		:
00004000	00232	U.266: .LONG	<<FPROT_VALUE-U.266>-4>	:
		:TPASMASK		:
0000*	00236	U.267: .LONG	16384	:
		:TPASTARGET		:
7044	00238	U.268: .WORD	<<WOPRO-U.268>-2>	:
		:TPASTYPE		:
00000000*	0023A	U.269: .WORD	28740	:
		:TPASADDR		:
00008000	0023E	U.270: .LONG	<<FPROT_VALUE-U.270>-4>	:
		:TPASMASK		:
0000*	00242	U.271: .LONG	32768	:
		:TPASTARGET		:
704C	00244	U.272: .WORD	<<WOPRO-U.272>-2>	:
		:TPASTYPE		:
00000000*	00246	U.273: .WORD	28748	:
		:TPASADDR		:
00008000	0024A	U.274: .LONG	<<FPROT_VALUE-U.274>-4>	:
		:TPASMASK		:
0000*	0024E	U.275: .LONG	32768	:
		:TPASTARGET		:
15F6	00250	U.276: .WORD	<<WOPRO-U.276>-2>	:
		:TPASTYPE		:
0000*	00252	U.277: .WORD	5622	:
		:TPASTARGET		:
	00254	U.278: .WORD	<<U.161-U.278>-2>	:
		:ENDPRO		:
102C	00254	U.161: .BLKB	0	:
		:TPASTYPE		:
0000*	00256	U.279: .WORD	4140	:
		:TPASTARGET		:
		U.280: .WORD	<<NEXTPRO-U.280>-2>	:

```
15F7 00258 :TPASTYPE  
          U.281: .WORD 5623  
FFFF 0025A :TPASTARGET  
          U.282: .WORD -1  
          0025C RET_STATE::  
          .BLKB 0  
          0025C RETSTART:  
          .BLKB 0  
4DF8 0025C :TPASTYPE  
          U.284: .WORD 19960  
0000* 0025E :TPASSUBEXP  
          U.286: .WORD <<U.285-U.286>-2>  
00000000* 00260 :TPASADDR  
          U.287: .LONG <<RETMIN_VALUE-U.287>-4>  
002C 00264 :TPASTYPE  
          U.288: .WORD 44  
15F7 00266 :TPASTYPE  
          U.289: .WORD 5623  
FFFF 00268 :TPASTARGET  
          U.290: .WORD -1  
4DF8 0026A :TPASTYPE  
          U.291: .WORD 19960  
0000* 0026C :TPASSUBEXP  
          U.292: .WORD <<U.285-U.292>-2>  
00000000* 0026E :TPASADDR  
          U.293: .LONG <<RETMAX_VALUE-U.293>-4>  
15F7 00272 :TPASTYPE  
          U.294: .WORD 5623  
FFFF 00274 :TPASTARGET  
          U.295: .WORD -1  
          00276 :GET DELTA  
          U.285: .BLKB 0  
91ED 00276 :TPASTYPE  
          U.296: .WORD -28179  
00000000V 00278 :TPASACTION  
          U.297: .LONG <<TEST_CHAR-U.297>-4>  
0000* 0027C :TPASTARGET  
          U.298: .WORD <<U.285-U.298>-2>  
15F6 0027E :TPASTYPE  
          U.299: .WORD 5622  
FFFF 00280 :TPASTARGET  
          U.300: .WORD -1  
  
          .PSECT _LIB$KEYOS,NOWRT, SHR, PIC,1  
  
00000 DC_KEYS::  
          .BLKB 0  
00000 :TPASKEY0  
          U.1: .BLKB 0  
0000* 00000 :TPASKEY  
          U.10: .WORD <U.9-U.1>  
0000* 00002 :TPASKEY  
          U.16: .WORD <U.15-U.1>  
0000* 00004 :TPASKEY  
          U.22: .WORD <U.21-U.1>  
0000* 00006 :TPASKEY  
          U.28: .WORD <U.27-U.1>
```

```
00008 OWNER_KEYS::
      :TPASKEY0 .BLKB 0
0000* 00008 U.38: .BLKB 0
      :TPASKEY
      U.40: .WORD <U.39-U.38>
0000A .BLKB 2
0000C JOURNAL_KEYS::
      :TPASKEY0 .BLKB 0
0000* 0000C U.58: .BLKB 0
      :TPASKEY
0000* 0000E U.60: .WORD <U.59-U.58>
      :TPASKEY
0000* 00010 U.66: .WORD <U.65-U.58>
      :TPASKEY
0000* 00012 U.72: .WORD <U.71-U.58>
      :TPASKEY
0000* 00014 U.78: .WORD <U.77-U.58>
      :TPASKEY
0000* 00016 U.84: .WORD <U.83-U.58>
      :TPASKEY
0000* 00018 U.90: .WORD <U.89-U.58>
      :TPASKEY
0000* 0001A U.96: .WORD <U.95-U.58>
      :TPASKEY
0000* 0001C U.102: .WORD <U.101-U.58>
      :TPASKEY
0000* 0001E U.108: .WORD <U.107-U.58>
      :TPASKEY
      U.114: .WORD <U.113-U.58>
00020 PRO_KEYS::
      :TPASKEY0 .BLKB 0
00020 U.124: .BLKB 0
0000* 00020 :TPASKEY
0000* 00022 U.126: .WORD <U.125-U.124>
      :TPASKEY
0000* 00024 U.134: .WORD <U.133-U.124>
      :TPASKEY
0000* 00026 U.142: .WORD <U.141-U.124>
      :TPASKEY
      U.150: .WORD <U.149-U.124>
00028 RET_KEYS::
      :TPASKEY0 .BLKB 0
00028 U.283: .BLKB 0
      .PSECT $OWNS,NOEXE,2
00000003 00000008 00000 TPARSE_BLOCK:
      :LONG 8 3
00008 .BLKB 28
00024 PRIVS: .BLKB 8
      QUAL_ACCESS== 1
      QUAL_BACKUP== 2
```

QUAL\_NOBACKUP== 3  
QUAL\_CONFIRM== 4  
QUAL\_DATA== 5  
QUAL\_ENTER== 6  
QUAL\_EOF== 7  
QUAL\_ERASE== 8  
QUAL\_NOERASE== 9  
QUAL\_EXPI== 10  
QUAL\_EXTE== 11  
QUAL\_FPROT== 12  
QUAL\_GBUF== 13  
QUAL\_JOURNAL== 14  
QUAL\_LABEL== 15  
QUAL\_LOG== 16  
QUAL\_NODI== 17  
QUAL\_OWNER== 18  
QUAL\_PARENT== 19  
QUAL\_REMOVE== 20  
QUAL\_RETENT== 21  
QUAL\_RPROT== 22  
QUAL\_TRUNC== 23  
QUAL\_USERNAME== 24  
QUAL\_VPROT== 25  
QUAL\_VRSN== 26  
QUAL\_WINDOWS== 27  
DATA\_READ== 1  
DATA\_NOREAD== 2  
DATA\_WRITE== 3  
DATA\_NOWRITE== 4  
JRNL\_AI== 1  
JRNL\_NOAI== 2  
JRNL\_AT== 3  
JRNL\_NOAT== 4  
JRNL\_BI== 5  
JRNL\_NOBI== 6  
JRNL\_RU== 7  
JRNL\_NORU== 8  
JRNL\_RUM== 9  
JRNL\_NORUM== 10

.EXTRN CALCULATE MAX, SYSSFAO  
.EXTRN LIBSTPARSE, LIBSCVT TIME  
.EXTRN LIBSCVT DTIME, LIBSCVT\_DTB  
.EXTRN RENAME BUF, FILE NAME  
.EXTRN FILE REF, SETSL STATUS  
.EXTRN SETSA CLIWORK, SETFILES\$FLAGS  
.EXTRN SETFICES\$D\$FLAGS, SETFILES\$J\$FLAGS  
.EXTRN ACC VALUE, EXP VALUE  
.EXTRN EXTE VALUE, FPROT VALUE  
.EXTRN GBUF VALUE, LABEL VALUE  
.EXTRN UIC VALUE, GROUP  
.EXTRN MEMBER, USER VALUE  
.EXTRN RETMIN VALUE, RETMAX VALUE  
.EXTRN VPROT VALUE, VRSN VALUE  
.EXTRN WINDOW VALUE, SETS FACILITY  
.EXTRN SETS OPERREQ, SETS\_WRITEERR  
.EXTRN SYSSSETPRV

		.PSECT		SCODE\$,NOWRT,2		
		003C	00000	.ENTRY	ACC_ACT, Save R2,R3,R4,R5	0396
55	00000000G	00	9E 00002	MOVAB	ACC_VALUE, R5	
54	00000000G	00	9E 00009	MOVAB	LIB\$STOP, R4	
5E	00000000'	08	C2 00010	SUBL2	#8, SP	
		EF	9F 00013	PUSHAB	PRIVS	0421
		01	DD 00019	PUSHL	#1	
7E		01	7D 0001B	MOVQ	#1, -(SP)	
00000000G		04	FB 0001E	CALLS	#4, SYSS\$SETPRV	
53		50	DD 00025	MOVL	R0, STATUS	
05		53	E8 00028	BLBS	STATUS, 1\$	
		53	DD 0002B	PUSHL	STATUS	0422
64		01	FB 0002D	CALLS	#1, LIB\$STOP	
09 00000000'		02	EO 00030	BBS	#2, PRIVS+2, 2\$	0424
	00000000G	8F	DD 00038	PUSHL	#SET\$ OPERR\$Q	
64		01	FB 0003E	CALLS	#1, LIB\$STOP	
65		03	DD 00041	MOVL	#3, ACC VALUE	0430
52	04	AC	DD 00044	MOVL	OPTION_BLOCK, R2	0435
	04	A2	B5 00048	TSTW	4(R2)	
		3C	13 0004B	BEQL	6\$	
		55	DD 0004C	PUSHL	R5	0441
	08	A2	DD 0004F	PUSHL	8(R2)	0442
	04	A2	3C 00052	MOVZWL	4(R2), -(SP)	0441
00000000G		03	FB 00056	CALLS	#3, LIB\$CVT_DTB	
53		50	DD 0005D	MOVL	R0, STATUS	
04		53	E8 00060	BLBS	STATUS, 3\$	
		53	DD 00063	PUSHL	STATUS	0447
		14	11 00065	BRB	5\$	0446
50		65	DD 00067	MOVL	ACC_VALUE, R0	0450
		09	19 0006A	BLSS	4\$	
000000FF		50	D1 0006C	CMPL	R0, #255	0452
		8F	15 00073	BLEQ	6\$	
	00000000*	8F	DD 00075	PUSHL	#<<<SET\$ FACILITY@16>+4584>+2>	0456
	04	A2	9F 0007B	PUSHAB	4(R2)	0455
		01	DD 0007E	PUSHL	#1	
	00000000*	8F	DD 00080	PUSHL	#<<<SET\$ FACILITY@16>+4344>+2>	
64		04	FB 00086	CALLS	#4, LIB\$STOP	
50		01	DD 00089	MOVL	#1, R0	0458
		04	0008C	RET		0459

; Routine Size: 141 bytes, Routine Base: \$CODE\$ + 0000

SETACT  
V04-000

M 3  
16-Sep-1984 01:06:01  
14-Sep-1984 12:08:59

VAX-11 Bliss-32 V4.0-742  
[CLIUTL.SRC]SETACT.B32;1

Page 30  
(13)

```
: 472      0460 1 GLOBAL ROUTINE back_act =  
: 473      0461 1 !++  
: 474      0462 1 !  
: 475      0463 1 ! This is the action routine for the /BACKUP qualifier. It simply  
: 476      0464 1 ! sets the correct bit in the flags word.  
: 477      0465 1 !  
: 478      0466 1 !--  
: 479      0467 2 BEGIN  
: 480      0468 2 setfile$flags[qual_backup] = true;  
: 481      0469 2 RETURN true;  
: 482      0470 1 END;
```

```
00000000G 00      0000 0000      .ENTRY BACK_ACT, Save nothing  
04 88 00002      BISB2 #4, SETFILE$FLAGS  
01 D0 00009      MOVL #1, R0  
04 0000C      RET
```

```
: 0460  
: 0468  
: 0469  
: 0470
```

; Routine Size: 13 bytes, Routine Base: \$CODE\$ + 008D



SETACT  
V04-000

N 3  
16-Sep-1984 01:06:01  
14-Sep-1984 12:08:59

VAX-11 Bliss-32 V4.0-742  
[CLIUTL.SRC]SETACT.B32;1

Page 31  
(14)

```
: 484      0471 1 GLOBAL ROUTINE noback_act =  
: 485      0472 1 |++  
: 486      0473 1 |  
: 487      0474 1 | This is the action routine for the /NOBACKUP qualifier. It simply  
: 488      0475 1 | sets the correct bit in the flags word.  
: 489      0476 1 |  
: 490      0477 1 |--  
: 491      0478 2 BEGIN  
: 492      0479 2 setfile$flags[qual_nobackup] = true;  
: 493      0480 2 RETURN true;  
: 494      0481 1 END;
```

```
00000000G 00      0000 0000  
08 88 00002  
01 D0 00009  
04 0000C
```

```
.ENTRY NOBACK ACT, Save nothing  
BISB2 #8, SETFILE$FLAGS  
MOVL #1, R0  
RET
```

```
: 0471  
: 0479  
: 0480  
: 0481
```

: Routine Size: 13 bytes, Routine Base: \$CODE\$ + 009A

```

496 0482 1 GLOBAL ROUTINE data_act (option_block,callback) =
497 0483 1 |++
498 0484 1 |
499 0485 1 | This is the action routine for the /DATA_CHECK qualifier. It checks to see
500 0486 1 | if any options were set. If not, it defaults to DATA_CHECK=WRITE.
501 0487 1 |
502 0488 1 |--
503 0489 2 BEGIN
504 0490 2
505 0491 2 LOCAL
506 0492 2     status;
507 0493 2
508 0494 2 MAP
509 0495 2     option_block : REF BBLOCK;
510 0496 2
511 0497 2 IF .option_block[cli$w_qdvalsiz] EQL 0
512 0498 2 THEN setfile$dflags[data_write] = true
513 0499 2
514 0500 2 ELSE
515 0501 2     BEGIN
516 0502 2         tparse_block[tpa$l_stringcnt] = .option_block[cli$w_qdvalsiz];
517 0503 2         tparse_block[tpa$l_stringptr] = .option_block[cli$a_qdvaladr];
518 0504 2         IF NOT (status = lib$tparse(tparse_block,
519 0505 2             dc_state,dc_keys))
520 0506 2     THEN
521 0507 2         BEGIN
522 0508 2             SIGNAL( set$_facility^16 + shr$_syntax + sts$_error,
523 0509 2                 1,
524 0510 2                 option_block[cli$q_qdvaldesc],
525 0511 2                 .status);
526 0512 2         RETURN .status;
527 0513 2     END;
528 0514 2 END;
529 0515 2 RETURN true;
530 0516 1 END;

```

			001C 0000	.ENTRY DATA ACT, Save R2,R3,R4	0482
54	00000000'	EF	9E 00002	MOVAB TPARSE_BLOCK+8, R4	0497
52	04	AC	D0 00009	MOVL OPTION_BLOCK, R2	
	04	A2	B5 0000D	TSTW 4(R2)	
		09	12 00010	BNEQ 1\$	
00000000G	00	08	88 00012	BISB2 #8, SETFILE\$DFLAGS	0498
		3D	11 00019	BRB 2\$	
	64	A2	3C 0001B 1\$:	MOVZWL 4(R2), TPARSE_BLOCK+8	0502
	04	A4	D0 0001F	MOVL 8(R2), TPARSE_BLOCK+12	0503
	00000000'	EF	9F 00024	PUSHAB DC_KEYS	0504
	00000000'	EF	9F 0002A	PUSHAB DC_STATE	
	F8	A4	9F 00030	PUSHAB TPARSE_BLOCK	
00000000G	00	03	FB 00033	CALLS #3, LIB\$TPARSE	
	53	50	D0 0003A	MOVL R0, STATUS	
	18	53	E8 0003D	BLBS STATUS, 2\$	
		53	DD 00040	PUSHL STATUS	0511
	04	A2	9F 00042	PUSHAB 4(R2)	0510

SETACT  
V04-000

<sup>C 4</sup>  
16-Sep-1984 01:06:01  
14-Sep-1984 12:08:59

VAX-11 Bliss-32 V4.0-742  
[CLIUTL.SRC]SETACT.B32;1

Page 33  
(15)

00000000G	00	00000000*	01 DD 00045	PUSHL #1
	50		8F DD 00047	PUSHL #<<<SETS FACILITY@16>+4344>+2>
			04 FB 0004D	CALLS #4, LIB\$SIGNAL
			53 D0 00054	MOVL STATUS, R0
			04 00057	RET
	50		01 D0 00058 2\$:	MOVL #1, R0
			04 0005B	RET

⋮  
⋮ 0512  
⋮  
⋮ 0515  
⋮ 0516

: Routine Size: 92 bytes, Routine Base: \$CODE\$ + 00A7

```

: 532      0517 1 GLOBAL ROUTINE enter_act (option_block,callback) =
: 533      0518 1 |++
: 534      0519 1 |
: 535      0520 1 | This is the action routine for the /ENTER qualifier.
: 536      0521 1 | The new synonym is collected.
: 537      0522 1 |
: 538      0523 1 |--
: 539      0524 2 BEGIN
: 540      0525 2
: 541      0526 2 MAP
: 542      0527 2     option_block : REF BBLOCK;
: 543      0528 2
: 544      0529 2
: 545      0530 2 |
: 546      0531 2 | Get the expanded file string
: 547      0532 2 |
: 548      0533 2
: 549      0534 2 CHSMOVE(.option_block[cli$w_qdvalsiz],           ! Move file string
: 550      0535 2     .option_block[cli$a_qdvaladr],
: 551      0536 2     rename_buf);
: 552      0537 2
: 553      0538 2 file_name[0] = .option_block[cli$w_qdvalsiz];      ! Store length
: 554      0539 2 file_name[1] = .option_block[cli$a_qdvaladr];    ! and address
: 555      0540 2
: 556      0541 2 RETURN true;
: 557      0542 1 END;

```

					007C 0000	.ENTRY	ENTER ACT, Save R2,R3,R4,R5,R6	: 0517
			56	04	AC D0 00002	MOVL	OPTION_BLOCK, R6	: 0534
00000000G	00	08	B6	04	A6 28 00006	MOVC3	4(R6), @8(R6), RENAME_BUF	
		00000000G	00	04	A6 3C 00010	MOVZWL	4(R6), FILE_NAME	: 0538
		00000000G	00	08	A6 D0 00018	MOVL	8(R6), FILE_NAME+4	: 0539
			50	01	D0 00020	MOVL	#1, R0	: 0541
					04 00023	RET		: 0542

: Routine Size: 36 bytes, Routine Base: \$CODE\$ + 0103

```

: 559      0543 1 GLOBAL ROUTINE erase_act =
: 560      0544 1 |++
: 561      0545 1 |
: 562      0546 1 | This is the action routine for the /ERASE qualifier. It simply
: 563      0547 1 | sets the correct bit in the flags word.
: 564      0548 1 |
: 565      0549 1 |--
: 566      0550 2 BEGIN
: 567      0551 2 setfile$flags[qual_erase] = true;
: 568      0552 2 RETURN true;
: 569      0553 1 END;

```

```

                                0000 0000      .ENTRY ERASE ACT, Save nothing      : 0543
                                01 88 00002     BISB2 #1, SETFILE$FLAGS+1      : 0551
                                01 D0 00009     MOVL #1, R0                    : 0552
                                04 0000C     RET                          : 0553

```

: Routine Size: 13 bytes, Routine Base: \$CODE\$ + 0127

```
: 571      0554 1 GLOBAL ROUTINE noerase_act =  
: 572      0555 1 |++  
: 573      0556 1 |  
: 574      0557 1 | This is the action routine for the /NOERASE qualifier. It simply  
: 575      0558 1 | sets the correct bit in the flags word.  
: 576      0559 1 |  
: 577      0560 1 |--  
: 578      0561 2 BEGIN  
: 579      0562 2 setfile$flags[qual_noerase] = true;  
: 580      0563 2 RETURN true;  
: 581      0564 1 END;
```

```
00000000G 00      0000 0000      .ENTRY NOERASE_ACT, Save nothing      : 0554  
02 88 00002      BISB2 #2, SETFILE$FLAGS+1      : 0562  
01 D0 00009      MOVL #1, R0      : 0563  
04 0000C      RET      : 0564
```

: Routine Size: 13 bytes, Routine Base: \$CODE\$ + 0134

```

583 0565 1 GLOBAL ROUTINE exp_act (option_block,callback) =
584 0566 1 |++
585 0567 1 |
586 0568 1 | This is the action routine for the /EXPIRATION qualifier.
587 0569 1 | If no value is given, exit with a syntax error.
588 0570 1 |
589 0571 1 |--
590 0572 1 BEGIN
591 0573 1
592 0574 1 LOCAL
593 0575 1     status,
594 0576 1     desc : BBLOCK[dsc$c_s_bln];
595 0577 1
596 0578 1 MAP
597 0579 1     option_block : REF BBLOCK;           ! Define the CLI options block
598 0580 1
599 0581 1 |
600 0582 1 | Get the date, signalling a syntax error if no good.
601 0583 1 |
602 0584 1
603 0585 1 desc[dsc$w_length] = .option_block[cli$q_dvalsiz];
604 0586 1 desc[dsc$a_pointer] = .option_block[cli$a_qdvaladr];
605 0587 1 IF NOT (status = LIB$CVT_TIME(desc,exp_value))
606 0588 1 THEN
607 0589 1     BEGIN
608 0590 1     SIGNAL_STOP( set$facility^16 + shr$s_syntax + sts$k_error,
609 0591 1                   1,
610 0592 1                   option_block[cli$q_qdvaldesc],
611 0593 1                   .status);
612 0594 1     RETURN .status;
613 0595 1     END
614 0596 1 ELSE RETURN true;
615 0597 1 END;

```

			000C 00000	.ENTRY	EXP_ACT, Save R2,R3	: 0565
	5E		08 C2 00002	SUBL2	#8, SP	: 0585
	52	04	AC D0 00005	MOVL	OPTION_BLOCK, R2	: 0586
	6E	04	A2 B0 00009	MOVW	4(R2), DESC	: 0587
	04	AE	A2 D0 0000D	MOVL	8(R2), DESC+4	
		00000000G	00 9F 00012	PUSHAB	EXP VALUE	
			04 AE 9F 00018	PUSHAB	DESC	
	00000000G	00	02 FB 0001B	CALLS	#2, LIB\$CVT_TIME	
	53		50 D0 00022	MOVL	R0, STATUS	
	18		53 E8 00025	BLBS	STATUS, 1\$	
			53 DD 00028	PUSHL	STATUS	: 0593
		04	A2 9F 0002A	PUSHAB	4(R2)	: 0592
			01 DD 0002D	PUSHL	#1	
	00000000G	00000000*	8F DD 0002F	PUSHL	#<<<SET\$ FACILITY@16>+4344>+2>	
	50		04 FB 00035	CALLS	#4, LIB\$STOP	: 0596
			53 D0 0003C	MOVL	STATUS, R0	
			04 0003F	RET		: 0597
	50		01 D0 00040 1\$:	MOVL	#1, R0	
			04 00043	RET		

SETACT  
V04-000

<sup>4</sup>  
~~16-Sep-1984~~ 01:06:01  
14-Sep-1984 12:08:59

VAX-11 Bliss-32 V4.0-742  
[CLIUTL.SRC]SETACT.B32,1

; Routine Size: 68 bytes, Routine Base: \$CODES + 0141



SETACT  
V04-000

1 4  
16-Sep-1984 01:06:01  
14-Sep-1984 12:08:59

VAX-11 Bliss-32 V4.0-742  
[CLIUTL.SRC]SETACT.B32;1

Page 77  
(

```
: 617      0598 1 GLOBAL ROUTINE noexp_act =  
: 618      0599 1 |++  
: 619      0600 1 |  
: 620      0601 1 | This is the action routine for the /NOEXPIRATION_DATE qualifier.  
: 621      0602 1 | It supplies an expiration date of zero.  
: 622      0603 1 |  
: 623      0604 1 |--  
: 624      0605 2 BEGIN  
: 625      0606 2  
: 626      0607 2 CHSFILL(0,8,exp_value);           ! Zero out the expiration date  
: 627      0608 2 setfile$flags[qual_expi] = true;      ! Set the expiration flag on  
: 628      0609 2  
: 629      0610 2 RETURN true;  
: 630      0611 1 END;
```

```
08          00          6E          003C 00000          .ENTRY NOEXP_ACT, Save R2,R3,R4,R5  
          00000000G 00 00000000G 00 2C 00002          MOVCS #0, (SP), #0, #8, EXP_VALUE          : 0598  
          00000000G 00 00000000G 04 88 0000C          BISB2 #4, SETFILE$FLAGS+1          : 0607  
          00000000G 50 00000000G 01 D0 00013          MOVL #1, R0          : 0608  
          00000000G 04 00016          RET          : 0610  
          : 0611
```

: Routine Size: 23 bytes, Routine Base: \$CODE\$ + 0185

```

632 0612 1 GLOBAL ROUTINE ext_act (option_block,callback) =
633 0613 1 |++
634 0614 1 |
635 0615 1 | This is the action routine for the /EXTENSION qualifier. If no value is
636 0616 1 | specified, the default value of 0 is used.
637 0617 1 |
638 0618 1 |--
639 0619 2 BEGIN
640 0620 2
641 0621 2 LOCAL
642 0622 2     status,
643 0623 2     desc : BBLOCK[dsc$c_s_bln];
644 0624 2
645 0625 2 MAP
646 0626 2     option_block : REF BBLOCK;           ! Define the CLI options block
647 0627 2
648 0628 2     exte_value = 0;                       ! Set up default
649 0629 2
650 0630 2 |
651 0631 2 | See if a value was specified. If not, then use the default.
652 0632 2 |
653 0633 2 IF .option_block[cli$w_qdvalsiz] EQL 0
654 0634 2 THEN RETURN true;
655 0635 2 |
656 0636 2 | If the value is there, convert it and return
657 0637 2 |
658 0638 2 |
659 0639 2 IF NOT (status = LIB$CVT_DTB(.option_block[cli$w_qdvalsiz],
660 0640 2     .option_block[cli$a_qdvaladr],
661 0641 2     exte_value))
662 0642 2 THEN
663 0643 2     BEGIN
664 0644 2     SIGNAL_STOP(set$_facility^16 + shr$_syntax + sts$k_error,
665 0645 2         1,
666 0646 2         option_block[cli$q_qdvaldesc],
667 0647 2         .status);
668 0648 2     END
669 0649 2 ELSE
670 0650 2     BEGIN
671 0651 2     IF NOT (.exte_value GEQ 0
672 0652 2         AND
673 0653 2         .exte_value LEQ 65535)
674 0654 2     THEN SIGNAL_STOP(set$_facility^16 + shr$_syntax + sts$k_error,
675 0655 2         1,
676 0656 2         option_block[cli$q_qdvaldesc],
677 0657 2         set$_facility^16 + shr$_valerr + sts$k_error);
678 0658 2     END;
679 0659 2
680 0660 2 RETURN true;
681 0661 1 END;

```

53 0000000G 00 000C 0000 .ENTRY EXT ACT, Save R2,R3  
00 9E 0002 MOVAB EXTE\_VALUE, R3

: 0612  
:

SETACT  
V04-000

K 4  
16-Sep-1984 01:06:01 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:08:59 [CLIUTL.SRC]SETACT.B32;1

Page 41  
(21)

5E		08	C2	00009	SUBL2	#8, SP		
		63	D4	0000C	CLRL	EXTE_VALUE		0628
52		AC	D0	0000E	MOVL	OPTION_BLOCK, R2		0633
	04	A2	B5	00012	TSTW	4(R2)		
	04	3D	13	00015	BEQL	4\$		
		53	DD	00017	PUSHL	R3		0639
	08	A2	DD	00019	PUSHL	8(R2)		0640
	04	A2	3C	0001C	MOVZWL	4(R2), -(SP)		0639
00000000G	7E	03	FB	00020	CALLS	#3, LIB\$CVT_DTB		
	00	50	E8	00027	BLBS	STATUS, 1\$		
	04	50	DD	0002A	PUSHL	STATUS		0647
		14	11	0002C	BRB	3\$		0646
	50	63	D0	0002E	MOVL	EXTE_VALUE, R0	1\$:	0651
		09	19	00031	BLSS	2\$		
0000FFFF	8F	50	D1	00033	CMPL	R0, #65535		0653
		18	15	0003A	BLEQ	4\$		
		8F	DD	0003C	PUSHL	#<<<SETS_FACILITY@16>+4584>+2>	2\$:	0657
	00000000*	A2	9F	00042	PUSHAB	4(R2)	3\$:	0656
	04	01	DD	00045	PUSHL	#1		
		8F	DD	00047	PUSHL	#<<<SETS_FACILITY@16>+4344>+2>		
00000000G	00	04	FB	0004D	CALLS	#4, LIB\$STOP		
	50	01	D0	00054	MOVL	#1, R0	4\$:	0660
		04	00057	RET				0661

; Routine Size: 88 bytes, Routine Base: \$CODE\$ + 019C

```

: 683 0662 1 GLOBAL ROUTINE fprot_act (option_block,callback) =
: 684 0663 1 !++
: 685 0664 1 !
: 686 0665 1 ! This is the action routine for the /FILE PROTECTION qualifier of
: 687 0666 1 ! SET VOLUME. The protection is parsed and stored away. If the
: 688 0667 1 ! protection is not valid, a fatal error message is issued.
: 689 0668 1 !
: 690 0669 1 !--
: 691 0670 2 BEGIN
: 692 0671 2
: 693 0672 2 LOCAL status; ! Status return
: 694 0673 2
: 695 0674 2 MAP option_block : REF BBLOCK; ! Define the option block
: 696 0675 2
: 697 0676 2 !
: 698 0677 2 ! Stuff the TPARSE block with the string
: 699 0678 2
: 700 0679 2 tparse_block[tpa$l_stringcnt] = .option_block[cli$w_qdvalsiz];
: 701 0680 2 tparse_block[tpa$l_stringptr] = .option_block[cli$a_qdvaladr];
: 702 0681 2
: 703 0682 2 fprot_value = 0; ! Initialize file protection value
: 704 0683 2
: 705 0684 2 !
: 706 0685 2 ! Now to parse the protection given. When finished, FPROT_VALUE will
: 707 0686 2 ! have the following values:
: 708 0687 2
: 709 0688 2 ! FPROT_VALUE[low_word] = protection value
: 710 0689 2 ! FPROT_VALUE[high_word] = group mask i.e. SYSTEM, OWNER, GROUP, WORLD
: 711 0690 2
: 712 0691 2 IF NOT (status = LIB$TPARSE(tparse_block,
: 713 0692 2 ! pro_state,
: 714 0693 2 ! pro_keys))
: 715 0694 2 THEN SIGNAL_STOP(set$_facility^T6 + shr$_syntax + sts$k_error,
: 716 0695 2 !
: 717 0696 2 ! option_block[cli$q_qdvaldesc],
: 718 0697 2 ! .status);
: 719 0698 2 RETURN true;
: 720 0699 1 END;

```

```

: 0662 .ENTRY FPROT_ACT, Save R2,R3
: 0679 MOVAB TPARSE_BLOCK+8, R3
: 0680 MOVL OPTION_BLOCK, R2
: 0682 MOVZWL 4(R2), TPARSE_BLOCK+8
: 0691 MOVL 8(R2), TPARSE_BLOCK+12
: CLRL FPROT_VALUE
: 0682 PUSHAB PRO_KEYS
: 0691 PUSHAB PRO_STATE
: PUSHAB TPARSE_BLOCK
: CALLS #3, LIB$TPARSE
: BLBS STATUS, 18
: 0697 PUSHL STATUS
: 0696 PUSHAB 4(R2)
: PUSHL #1

```

SETACT  
V04-000

M 4  
16-Sep-1984 01:06:01 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:08:59 [CLIUTL.SRC]SETACT.B32;1

Page 43  
(22)

00000000G	00	00000000*	8F	DD	0003C	PUSHL	#<<<SETS FACILITY@16>+4344>+2>
	50		04	FB	00042	CALLS	#4, LIB\$STOP
			01	DO	00049	MOVL	#1, R0
			04	0004C		RET	

:  
:  
: 0698  
: 0699

: Routine Size: 77 bytes, Routine Base: \$CODE\$ + 01F4

```

: 722 0700 1 GLOBAL ROUTINE gbuf_act (option_block,callback) =
: 723 0701 1 ++
: 724 0702 1 |
: 725 0703 1 | This is the action routine for the GLOBAL_BUFFER qualifier. The number of
: 726 0704 1 | global buffers desired is collected.
: 727 0705 1 |
: 728 0706 1 | --
: 729 0707 2 BEGIN
: 730 0708 2
: 731 0709 2 LOCAL
: 732 0710 2     status,
: 733 0711 2     desc : BBLOCK[dsc$c_s_bln];
: 734 0712 2
: 735 0713 2 MAP
: 736 0714 2     option_block : REF BBLOCK;           ! Define the CLI options block
: 737 0715 2
: 738 0716 2 |
: 739 0717 2 | Convert the value given (in ASCII) to a numeric value.
: 740 0718 2 |
: 741 0719 2 IF NOT (status = LIB$CVT_DTB(.option_block[cli$q_dvalsiz],
: 742 0720 2     .option_block[cli$a_qdvaladr],
: 743 0721 2     gbuf_value))
: 744 0722 2 THEN
: 745 0723 2     BEGIN
: 746 0724 2     SIGNAL_STOP(set$_facility^16 + shr$_syntax + sts$k_error,
: 747 0725 2     1,
: 748 0726 2     option_block[cli$q_qdvaldesc],
: 749 0727 2     .status);
: 750 0728 2     END
: 751 0729 2 ELSE
: 752 0730 2     BEGIN
: 753 0731 2 |
: 754 0732 2 | If the value is not a word or less in length, signal an error.
: 755 0733 2 |
: 756 0734 2     IF NOT (.gbuf_value GEQ 0 AND .gbuf_value LEQ 65535)
: 757 0735 2     THEN SIGNAL_STOP(set$_facility^16 + shr$_syntax + sts$k_error,
: 758 0736 2     1,
: 759 0737 2     option_block[cli$q_qdvaldesc],
: 760 0738 2     set$_facility^16 + shr$_valerr + sts$k_error);
: 761 0739 2     END;
: 762 0740 2
: 763 0741 2 RETURN true;
: 764 0742 1 END;

```

			000C 00000	.ENTRY	GBUF_ACT, Save R2,R3	
53	00000000G	00	9E 00002	MOVAB	GBUF_VALUE, R3	0700
5E		08	C2 00009	SUBL2	#8, SP	
		53	DD 0000C	PUSHL	R3	0719
52	04	AC	DD 0000E	MOVL	OPTION_BLOCK, R2	0720
		08	DD 00012	PUSHL	8(R2)	
7E	04	A2	3C 00015	MOVZWL	4(R2), -(SP)	0719
00000000G	00	03	FB 00019	CALLS	#3, LIB\$CVT_DTB	
	04	50	E8 00020	BLBS	STATUS, 1\$	

		50	DD	00023	PUSHL	STATUS	:	0727		
		14	11	00025	BRB	3\$	:	0726		
	50	63	D0	00027	1\$:	MOVL	GBUF_VALUE, R0	:	0734	
		09	19	0002A	BLSS	2\$	:			
0000FFFF	8F	50	D1	0002C	CMPL	R0, #65535	:			
		18	15	00033	BLEQ	4\$	:			
		00000000*	8F	DD	00035	2\$:	PUSHL	#<<<SETS_FACILITY@16>+4584>+2>	:	0738
		04	A2	9F	0003B	3\$:	PUSHAB	4(R2)	:	0737
			01	DD	0003E	PUSHL	#1	:		
		00000000*	8F	DD	00040	PUSHL	#<<<SETS_FACILITY@16>+4344>+2>	:		
00000000G	00	04	FB	00046	CALLS	#4, LIB\$STOP	:			
	50	01	D0	0004D	4\$:	MOVL	#1, R0	:	0741	
			04	00050	RET		:	0742		

; Routine Size: 81 bytes, Routine Base: \$CODE\$ + 0241

```

: 766 0743 1 GLOBAL ROUTINE journal_act (option_block,callback) =
: 767 0744 1 |++
: 768 0745 1 |
: 769 0746 1 | This is the action routine for the /JOURNAL qualifier. Based on the
: 770 0747 1 | journal types set, specific journaling bits are set.
: 771 0748 1 |
: 772 0749 1 |--
: 773 0750 2 BEGIN
: 774 0751 2
: 775 0752 2 LOCAL
: 776 0753 2     status;
: 777 0754 2
: 778 0755 2 MAP
: 779 0756 2     option_block : REF BBLOCK;
: 780 0757 2
: 781 0758 2 |
: 782 0759 2 | Use TPARSE to parse the journal list.
: 783 0760 2 |
: 784 0761 2
: 785 0762 2 tparse_block[tpa$l_stringcnt] = .option_block[cli$q_dvalsiz];
: 786 0763 2 tparse_block[tpa$l_stringptr] = .option_block[cli$a_qvaladr];
: 787 0764 2
: 788 0765 3 IF NOT (status = LIB$TPARSE(tparse_block, journal_state, journal_keys))
: 789 0766 2 THEN SIGNAL_STOP(set$_facility^16 + shr$_syntax + sts$k_error,
: 790 0767 2     1,
: 791 0768 2     option_block[cli$q_qvaldesc],
: 792 0769 2     .status);
: 793 0770 2 |
: 794 0771 2 | If both RU and RUM were specified, then signal a syntax error.
: 795 0772 2 |
: 796 0773 2
: 797 0774 3 IF (.setfile$jflags[jrnl_ru] AND .setfile$jflags[jrnl_rum])
: 798 0775 2 THEN SIGNAL_STOP(set$_facility^16 + shr$_syntax + sts$k_error,
: 799 0776 2     1,
: 800 0777 2     option_block[cli$q_qvaldesc],
: 801 0778 2     set$_facility^16 + shr$_confqual + sts$k_error);
: 802 0779 2
: 803 0780 2 RETURN true;
: 804 0781 1 END;

```

			001C 0000	.ENTRY	JOURNAL_ACT, Save R2,R3,R4	: 0743
	54	00000000G	00 9E 00002	MOVAB	LIB\$STOP, R4	:
	53	00000000'	EF 9E 00009	MOVAB	TPARSE_BLOCK+8, R3	:
	52	04	AC D0 00010	MOVL	OPTION_BLOCK, R2	: 0762
	63	04	A2 3C 00014	MOVZWL	4(R2), TPARSE_BLOCK+8	:
04	A3	08	A2 D0 00018	MOVL	8(R2), TPARSE_BLOCK+12	: 0763
		00000000'	EF 9F 0001D	PUSHAB	JOURNAL_KEYS	: 0765
		00000000'	EF 9F 00023	PUSHAB	JOURNAL_STATE	:
		F8	A3 9F 00029	PUSHAB	TPARSE_BLOCK	:
00000000G	00		03 FB 0002C	CALLS	#3, LIB\$TPARSE	:
	10		50 E8 00033	BLBS	STATUS, 1\$	:
			50 DD 00036	PUSHL	STATUS	: 0769
		04	A2 9F 00038	PUSHAB	4(R2)	: 0768



SETACT  
V04-000

D 5  
16-Sep-1984 01:06:01  
14-Sep-1984 12:08:59

VAX-11 Bliss-32 V4.0-742  
[CLIUTL.SRC]SETACT.B32;1

Page 47  
(24)

		01	DD	0003B		PUSHL	#1		
		8F	DD	0003D		PUSHL	#<<<SET\$ FACILITY@16>+4344>+2>		
64	00000000*	04	FB	00043		CALLS	#4, LIB\$STOP		
	00000000G	00	95	00046	1\$:	TSTB	SETFILE\$JFLAGS		0774
		1C	18	0004C		BGEQ	2\$		
14	00000000G	00	01	E1	0004E	BBC	#1, SETFILE\$JFLAGS+1, 2\$		
	00000000*	8F	DD	00056		PUSHL	#<<<SET\$ _FACILITY@16>+4832>+2>		0778
		04	A2	9F	0005C	PUSHAB	4(R?)		0777
		01	DD	0005F		PUSHL	#1		
	00000000*	8F	DD	00061		PUSHL	#<<<SET\$ FACILITY@16>+4344>+2>		
64		04	FB	00067		CALLS	#4, LIB\$STOP		
50		01	D0	0006A	2\$:	MOVL	#1, R0		0780
		04	00	0006D		RET			0781

: Routine Size: 110 bytes, Routine Base: \$CODE\$ + 0292

```

: 806 0782 1 GLOBAL ROUTINE label_act (option_block, callback) =
: 807 0783 1 |++
: 808 0784 1 |
: 809 0785 1 | This is the action routine for the LABEL qualifier of SET VOLUME. It
: 810 0786 1 | retrieves the value of the string, checks that it is no longer than
: 811 0787 1 | twelve characters, and stores length and location in LABEL_VALUE.
: 812 0788 1 |
: 813 0789 1 |--
: 814 0790 2 BEGIN
: 815 0791 2
: 816 0792 2 LOCAL status; ! Status return
: 817 0793 2
: 818 0794 2 MAP option_block : REF BBLOCK; ! Define the cli block
: 819 0795 2
: 820 0796 2 |
: 821 0797 2 | Check that the string is no longer than twelve characters.
: 822 0798 2
: 823 0799 2 IF .option_block[cli$w_qdvalsiz] GTR 12
: 824 0800 2 THEN SIGNAL_STOP(set$_facility^16 + shr$_syntax + sts$k_error,
: 825 0801 2 1,
: 826 0802 2 option_block[cli$q_qdvaldesc],
: 827 0803 2 set$_facility^16 + shr$_valerr + sts$k_error);
: 828 0804 2 |
: 829 0805 2 | Store the location and length in LABEL_VALUE
: 830 0806 2
: 831 0807 2 label_value[0] = .option_block[cli$w_qdvalsiz];
: 832 0808 2 label_value[1] = .option_block[cli$a_qdvaladr];
: 833 0809 2
: 834 0810 2 RETURN true;
: 835 0811 1 END;

```

				0004 0000	.ENTRY LABEL ACT, Save R2	: 0782
	52	04	AC	D0 00002	MOVL OPTION_BLOCK, R2	: 0799
	0C	04	A2	B1 00006	CMPW 4(R2), #12	
				18 1B 0000A	BLEQU 1\$	
		00000000*	8F	DD 0000C	PUSHL #<<<SET\$_FACILITY@16>+4584>+2>	: 0803
			04	A2 9F 00012	PUSHAB 4(R2)	: 0802
				01 DD 00015	PUSHL #1	
		00000000*	8F	DD 00017	PUSHL #<<<SET\$_FACILITY@16>+4344>+2>	
	00000000G	00	04	FB 0001D	CALLS #4, LIB\$STOP	
	00000000G	00	04	A2 3C 00024	MOVZWL 4(R2), LABEL_VALUE	: 0807
	00000000G	00	08	A2 D0 0002C	MOVI 8(R2), LABEL_VALUE+4	: 0808
		50	01	D0 00034	MO #1, R0	: 0810
			04	00037	RET	: 0811

: Routine Size: 56 bytes, Routine Base: \$CODE\$ + 0300

```

837 0812 1 GLOBAL ROUTINE owner_act (option_block,callback) =
838 0813 1 |++
839 0814 1 |
840 0815 1 | This is the action routine for the OWNER_UIC qualifier. The input is
841 0816 1 | parsed to obtain the group and member numbers of the UIC.
842 0817 1 |
843 0818 1 |--
844 0819 2 BEGIN
845 0820 2
846 0821 2 LOCAL
847 0822 2     status;                ! Status
848 0823 2
849 0824 2 MAP
850 0825 2     option_block : REF BBLOCK;
851 0826 2
852 0827 2 uic_value = 0;          ! Set the UIC value to zero initially
853 0828 2
854 0829 2 |
855 0830 2 | Check to see if UIC specified. If not, use current process UIC.
856 0831 2 |
857 0832 2 IF .option_block[cli$qdvalsiz] EQL 0
858 P 0833 2 THEN $GETJPI(ITMLST = DPLIT(WORD(4,jpi$ uic),
859 P 0834 2     uic_value,
860 P 0835 2     0,
861 P 0836 2     0))
862 0837 2 ELSE
863 0838 2 BEGIN:
864 0839 3     *parse_block[tpa$l_stringcnt] = .option_block[cli$qdvalsiz];
865 0840 3     tparse_block[tpa$l_stringptr] = .option_block[cli$a_qdvaladr];
866 0841 4     IF NOT (status = lib$tparse(tparse_block,
867 0842 4         owner_state,
868 0843 4         owner_keys))
869 0844 3 THEN SIGNAL_STOP(set$_facility^16 + shr$_syntax + sts$k_error,
870 0845 3     1,
871 0846 3     option_block[cli$q_qdvaldesc]);
872 0847 3 IF NOT .setfile$flags[qual_parent]
873 0848 3 THEN
874 0849 4 BEGIN
875 0850 6     IF NOT ((.group LEQ %0'377' AND .group GEQ 0)
876 0851 5     AND
877 0852 5     (.member LEQ %0'377' AND .member GEQ 0))
878 0853 4 THEN SIGNAL_STOP(set$_facility^16 + shr$_syntax + sts$k_error,
879 0854 4     1,
880 0855 4     option_block[cli$q_qdvaldesc],
881 0856 4     set$_facility^16 + shr$_valerr + sts$k_error)
882 0857 4 ELSE uic_value = .group^16 + .member;
883 0858 3 END;
884 0859 2 END;
885 0860 2
886 0861 2 RETURN true;
887 0862 1 END;

```

.PSECT \$SPLITS,NOWRT,NOEXE,2

0304 0004 00000 P.AAA: .WORD 4, 772

				.ADDRESS	UIC	VALUE	
	00000000	00000000	00004	.LONG	0,	0	:
				.EXTRN	SYSSGETJPI		:
				.PSECT	\$CODE\$,NOWRT,2		:
		007C	00000	.ENTRY	OWNER ACT, Save R2,R3,R4,R5,R6		0812
56	00000000G	00	9E 00002	MOVAB	UIC VALUE, R6		:
55	00000000G	00	9E 00009	MOVAB	MEMBER, R5		:
54	00000000G	00	9E 00010	MOVAB	LIB\$STOP, R4		:
53	00000000'	EF	9E 00017	MOVAB	TPARSE_BLOCK+8, R3		:
		66	D4 0001E	CLRL	UIC VALUE		0827
52	04	AC	D0 00020	MOVL	OPTION_BLOCK, R2		0832
		04	A2 B5 00024	TSTW	4(R2)		:
		17	12 00027	BNEQ	1\$		:
		7E	7C 00029	CLRQ	-(SP)		0836
		7E	D4 0002B	CLRL	-(SP)		:
	00000000'	EF	9F 0002D	PUSHAB	P.AAA		:
		7E	7C 00033	CLRQ	-(SP)		:
		7E	D4 00035	CLRL	-(SP)		:
	00000000G	00	07 FB 00037	CALLS	#7, SYSSGETJPI		:
		7A	11 0003E	BRB	5\$		:
	04	63	A2 3C 00040	MOVZWL	4(R2), TPARSE_BLOCK+8		0839
		A3	A2 D0 00044	MOVL	8(R2), TPARSE_BLOCK+12		0840
			EF 9F 00049	PUSHAB	OWNER_KEYS		0841
			EF 9F 0004F	PUSHAB	OWNER_STATE		:
			F8 A3 9F 00055	PUSHAB	TPARSE_BLOCK		:
	00000000G	00	03 FB 00058	CALLS	#3, LIB\$TPARSE		:
		0E	50 E8 0005F	BLBS	STATUS, 2\$		:
		04	A2 9F 00062	PUSHAB	4(R2)		0846
			01 DD 00065	PUSHL	#1		:
			8F DD 00067	PUSHL	#<<<SETS FACILITY@16>+4344>+2>		:
		64	03 FB 0006D	CALLS	#3, LIB\$STOP		:
42	00000000G	00	03 E0 00070	BBS	#3, SETFILES\$FLAGS+2, 5\$		0847
		50	00 D0 00078	MOVL	GROUP, R0		0850
	000000FF	8F	50 D1 0007F	CMP	R0, #255		:
			14 14 00086	BGTR	3\$		:
			50 D5 00088	TSTL	R0		:
			10 19 0008A	BLSS	3\$		:
		51	65 D0 0008C	MOVL	MEMBER, R1		0852
	000000FF	8F	51 D1 0008F	CMP	R1, #255		:
			04 14 00096	BGTR	3\$		:
			51 D5 00098	TSTL	R1		:
			16 18 0009A	BGEQ	4\$		:
			8F DD 0009C	PUSHL	#<<<SETS FACILITY@16>+4584>+2>		0856
		04	A2 9F 000A2	PUSHAB	4(R2)		0855
			01 DD 000A5	PUSHL	#1		:
			8F DD 000A7	PUSHL	#<<<SETS FACILITY@16>+4344>+2>		:
		64	04 FB 000AD	CALLS	#4, LIB\$STOP		:
			08 11 000B0	BRB	5\$		:
	50	50	10 78 000B2	ASHL	#16, R0, R0		0857
	66	50	65 C1 000B6	ADDL3	MEMBER, R0, UIC_VALUE		:
		50	01 D0 000BA	MOVL	#1, R0		0861
			04 000BD	RET			0862

; Routine Size: 190 bytes, Routine Base: \$CODE\$ + 0338

SETACT  
V04-000

H 5  
16-Sep-1984 01:06:01  
14-Sep-1984 12:08:59

VAX-11 Bliss-32 V4.0-742  
[CLIUTL.SRC]SETACT.B32;1

Page 51  
(26)

```

889 0863 1 GLOBAL ROUTINE retent_act (option_block,callback) =
890 0864 1 ++
891 0865 1
892 0866 1 This is the action routine for the /RETENTION qualifier.
893 0867 1 The minimum retention value must be given. If no maximum retention value is
894 0868 1 specified, a value of twice the minimum (but no more than a week more than
895 0869 1 the minimum) is used.
896 0870 1
897 0871 1 --
898 0872 2 BEGIN
899 0873 2
900 0874 2 LOCAL
901 0875 2     status,
902 0876 2     temp_desc : BBLOCK[dsc$sc_s_bln];
903 0877 2
904 0878 2 MAP
905 0879 2     option_block : REF BBLOCK;           ! Define the CLI options block
906 0880 2
907 0881 2
908 0882 2
909 0883 2 Parse the input, to obtain the minimum and maximum retention times.
910 0884 2
911 0885 2
912 0886 2 CH$FILL(0, 8, retmin_value);           ! Zero minimum value
913 0887 2 CH$FILL(0, 8, retmax_value);         ! Zero maximum value
914 0888 2
915 0889 2 tparse_block[tpa$l_stringcnt] = .option_block[cli$w_qdvalsiz];
916 0890 2 tparse_block[tpa$l_stringptr] = .option_block[cli$a_qdvaladr];
917 0891 2 IF NOT (status = lib$tparse(tparse_block, ret_state, ret_keys))
918 0892 2 THEN
919 0893 2     BEGIN
920 0894 2         SIGNAL(set$_facility^16 + shr$_syntax + sts$_k_error,
921 0895 2             1,
922 0896 2             option_block[cli$q_qdvaldesc]);
923 0897 2     RETURN false;           ! If error in parse, return false
924 0898 2     END;
925 0899 2
926 0900 2
927 0901 2 Parse the input, to obtain the minimum and maximum retention times.
928 0902 2
929 0903 2
930 0904 2 IF .retmin_value[0] EQL 0
931 0905 2 THEN
932 0906 2     BEGIN
933 0907 2         SIGNAL(set$_facility^16 + shr$_syntax + sts$_k_error,
934 0908 2             1,
935 0909 2             option_block[cli$q_qdvaldesc]);
936 0910 2     RETURN false;
937 0911 2     END;
938 0912 2
939 0913 2
940 0914 2 Convert the minimum retention value to 64-bit system delta time format
941 0915 2
942 0916 2 IF NOT (status = LIB$CVT_DTIME(retmin_value, temp_desc))
943 0917 2 THEN
944 0918 2     BEGIN
945 0919 2         SIGNAL(set$_facility^16 + shr$_syntax + sts$_k_error,

```

```

: 946 0920 1,
: 947 0921 retmin_value,
: 948 0922 .status);
: 949 0923 RETURN false;
: 950 0924 END
: 951 0925 ELSE CH$MOVE(8, temp_desc, retmin_value); ! If no error, put 64-bit
: 952 0926 ! delta time in place
: 953 0927
: 954 0928
: 955 0929 ! If a maximum value was supplied, then convert it in the same way.
: 956 0930
: 957 0931
: 958 0932 IF .retmax_value[0] NEQ 0
: 959 0933 THEN
: 960 0934 BEGIN
: 961 0935 IF NOT (status = LIB$CVT_DTIME(retmax_value, temp_desc))
: 962 0936 THEN
: 963 0937 BEGIN
: 964 0938 SIGNAL(set$_facility^16 + shr$_syntax + sts$_error,
: 965 0939 1,
: 966 0940 retmax_value,
: 967 0941 .status);
: 968 0942 RETURN .status;
: 969 0943 END
: 970 0944 ELSE CH$MOVE(8, temp_desc, retmax_value);
: 971 0945 END
: 972 0946
: 973 0947 !
: 974 0948 ! If no maximum value was supplied, then use twice the minimum value. If this
: 975 0949 ! value is greater than a week, use only a week.
: 976 0950 !
: 977 0951
: 978 0952 ELSE calculate_max(retmin_value, retmax_value);
: 979 0953
: 980 0954 RETURN true;
: 981 0955 1 END;

```

			OFFC 00000		.ENTRY RETENT ACT, Save R2,R3,R4,R5,R6,R7,R8,R9,-	0863	
					R10,R11		
		5B	00000000G	00	9E 00002	MOVAB LIB\$CVT_DTIME, R11	
		5A	00000000G	00	9E 00009	MOVAB LIB\$SIGNAL, R10	
		59	00000000'	EF	9E 00010	MOVAB TPARSE_BLOCK+8, R9	
		58	00000000G	00	9E 00017	MOVAB RETMIN_VALUE, R8	
		57	00000000G	00	9E 0001E	MOVAB RETMAX_VALUE, R7	
		5E		08	C2 00025	SUBL2 #8, SP	
08	00	6E		00	2C 00028	MOVCS #0, (SP), #0, #8, RETMIN_VALUE	0886
				68	0002D		
08	00	6E		00	2C 0002E	MOVCS #0, (SP), #0, #8, RETMAX_VALUE	0887
				67	00033		
		52	04	AC	DO 00034	MOVL OPTION_BLOCK, R2	0889
		69	04	A2	3C 00038	MOVZWL 4(R2), TPARSE_BLOCK+8	
	04	A9	08	A2	DO 0003C	MOVL 8(R2), TPARSE_BLOCK+12	0890
			00000000'	EF	9F 00041	PUSHAB RET_KEYS	0891

		00000000'	EF	9F	00047	PUSHAB	RET STATE		
		FB	A9	9F	0004D	PUSHAB	TPARSE BLOCK		
00000000G	00		03	FB	00050	CALLS	#3, LIB\$TPARSE		
	56		50	D0	00057	MOVL	R0, STATUS		
	04		56	E9	0005A	BLBC	STATUS, 1\$		
			68	D5	0005D	TSTL	RETMIN_VALUE		0904
		04	10	12	0005F	BNEQ	2\$		
			A2	9F	00061	1\$: PUSHAB	4(R2)		0909
			01	DD	00064	PUSHL	#1		
		00000000*	8F	DD	00066	PUSHL	#<<<SET\$ FACILITY@16>+4344>+2>		
6A			03	FB	0006C	CALLS	#3, LIB\$SIGNAL		
			5B	11	0006F	BRB	7\$		0910
		4100	8F	BB	00071	2\$: PUSHR	#*M<R8, SP>		0916
6B			02	FB	00075	CALLS	#2, LIB\$CVT_DTIME		
	56		50	D0	00078	MOVL	R0, STATUS		
	11		56	E8	0007B	BLBS	STATUS, 3\$		
			56	DD	0007E	PUSHL	STATUS		0922
			58	DD	00080	PUSHL	R8		0919
			01	DD	00082	PUSHL	#1		
		00000000*	8F	DD	00084	PUSHL	#<<<SET\$ FACILITY@16>+4344>+2>		
6A			04	FB	0008A	CALLS	#4, LIB\$SIGNAL		
			3D	11	0008D	BRB	7\$		0923
68			08	28	0008F	3\$: MOV C3	#8, TEMP_DESC, RETMIN_VALUE		0925
			67	D5	00093	TSTL	RETMAX_VALUE		0932
			26	13	00095	BEQL	5\$		
		4080	8F	BB	00097	PUSHR	#*M<R7, SP>		0935
6B			02	FB	0009B	CALLS	#2, LIB\$CVT_DTIME		
	56		50	D0	0009E	MOVL	R0, STATUS		
	13		56	E8	000A1	BLBS	STATUS, 4\$		
			56	DD	000A4	PUSHL	STATUS		0941
			57	DD	000A6	PUSHL	R7		0938
			01	DD	000A8	PUSHL	#1		
		00000000*	8F	DD	000AA	PUSHL	#<<<SET\$ FACILITY@16>+4344>+2>		
6A			04	FB	000B0	CALLS	#4, LIB\$SIGNAL		
	50		56	D0	000B3	MOVL	STATUS, R0		0942
			04	000B6	RET				
67			08	28	000B7	4\$: MOV C3	#8, TEMP_DESC, RETMAX_VALUE		0944
			0B	11	000BB	BRB	6\$		0932
			57	DD	000BD	5\$: PUSHL	R7		0952
			58	DD	000BF	PUSHL	R8		
00000000G	00		02	FB	000C1	CALLS	#2, CALCULATE_MAX		
	50		01	D0	000C8	6\$: MOVL	#1, R0		0954
			04	000CB	RET				
			50	D4	000CC	7\$: CLRL	R0		0955
			04	000CE	RET				

; Routine Size: 207 bytes, Routine Base: \$CODE\$ + 03F6



```

: 983      0956 1 ROUTINE test_char =
: 984      0957 1 !++
: 985      0958 1
: 986      0959 1 This routine is used by TPARSE to process the /RETENTION values.
: 987      0960 1
: 988      0961 1 NOTE that this routine references the Argument Pointer (AP) directly,
: 989      0962 1 due to the fact that TPARSE does not follow the calling standard.
: 990      0963 1
: 991      0964 1 !--
: 992      0965 1
: 993      0966 2 BEGIN
: 994      0967 2
: 995      0968 2 BUILTIN AP;                ! Declare the argument pointer
: 996      0969 2
: 997      0970 2 LOCAL
: 998      0971 2     ptr,
: 999      0972 2     char : BYTE;
1000      0973 2
1001      0974 2 ptr = .ap + $BYTEOFFSET(tpa$b_char);    ! Look at the character just read
1002      0975 2 char = CHSRCHAR(.ptr);
1003      0976 2
1004      0977 2 IF .char EQL ','
1005      0978 2 THEN RETURN false                ! If a comma, then end of string
1006      0979 2 ELSE RETURN true;              ! Else continue processing
1007      0980 2
1008      0981 1 END;

```

```

                                0000 0000 TEST_CHAR:
                                .WORD Save nothing
                                MOVAB 24(AP), PTR                : 0956
                                MOVB (PTR), CHAR                  : 0974
                                CMPB CHAR, #44                   : 0975
                                BNEQ 1$                          : 0977
                                CLRL R0
                                RET
                                50                                04 00010
                                50                                01 00 00011 1$: MOVL #1, R0
                                04 00014                          RET
                                                                : 0979
                                                                : 0981

```

: Routine Size: 21 bytes, Routine Base: \$RODES + 04C5

```

: 1010      0982 1 GLOBAL ROUTINE user_act (option_block, callback) =
: 1011      0983 1 |++
: 1012      0984 1 |
: 1013      0985 1 | This is the action routine for the USER_NAME qualifier of SET VOLUME. It
: 1014      0986 1 | retrieves the value of the string, checks that it is no longer than
: 1015      0987 1 | twelve characters, and stores a descriptor pointing to it.
: 1016      0988 1 |
: 1017      0989 1 |--
: 1018      0990 2 BEGIN
: 1019      0991 2
: 1020      0992 2 OWN user_label : VECTOR[12,BYTE];          ! Place to put process username
: 1021      0993 2
: 1022      0994 2 LOCAL status;                          ! Status return
: 1023      0995 2
: 1024      0996 2 MAP option_block : REF BBLOCK;          ! Define the cli block
: 1025      0997 2
: 1026      0998 2 |
: 1027      0999 2 | If no username was specified, use the current process username.
: 1028      1000 2
: 1029      1001 2 IF .option_block[cli$qdvalsiz] EQL 0
: 1030      1002 2 THEN
: 1031      1003 2 BEGIN
: 1032      P 1004 2 $GETJPI(ITMLST = UPLIT(WORD(4, jpi$username),          ! Get the username
: 1033      P 1005 2 user_label,                                          ! Store it here
: 1034      P 1006 2 user_value[0],                                       ! Store length here
: 1035      1007 2 0));
: 1036      1008 2 user_value[1] = user_label;
: 1037      1009 2 END
: 1038      1010 2
: 1039      1011 2 ELSE
: 1040      1012 2 BEGIN
: 1041      1013 2
: 1042      1014 2 |
: 1043      1015 2 | Check that the string is no longer than twelve characters.
: 1044      1016 2 |
: 1045      1017 2 | IF .option_block[cli$qdvalsiz] GTR 12
: 1046      1018 2 | THEN SIGNAL_STOP(set$facility^16 + shr$syntax + sts$k_error,
: 1047      1019 2 | 1,
: 1048      1020 2 | option_block[cli$qdvaldesc],
: 1049      1021 2 | set$facility^16 + shr$valerr + sts$k_error);
: 1050      1022 2 |
: 1051      1023 2 | Record the length and location in USER_VALUE.
: 1052      1024 2 |
: 1053      1025 2 | user_value[0] = .option_block[cli$qdvalsiz];
: 1054      1026 2 | user_value[1] = .option_block[cli$a_qdvaladr];
: 1055      1027 2 | END;
: 1056      1028 2
: 1057      1029 2 RETURN true;
: 1058      1030 1 END;

```

```

.PSECT $PLITS,NOWRT,NOEXE,2
0202 0004 00010 P.AAB: .WORD 4, 514
00000000G 00000000' 00014 .ADDRESS USER_LABEL, USER_VALUE
00000000 0001C .LONG 0

```

.PSECT \$OWNS,NOEXE,2

0002C USER\_LABEL:  
.BLKB 12

.PSECT \$CODE\$,NOWRT,2

		000C 0000	.ENTRY	USER_ACT, Save R2,R3	: 0982
53	00000000G	00 9E 00002	MOVAB	USER_VALUE+4, R3	: 1001
52	04	AC D0 00009	MOVL	OPTION_BLOCK, R2	: 1007
	04	A2 B5 0000D	TSTW	4(R2)	: 1008
		1E 12 00010	BNEQ	1\$	: 1009
		7E 7C 00012	CLRQ	-(SP)	: 1010
		7E D4 00014	CLRL	-(SP)	: 1011
	00000000'	EF 9F 00016	PUSHAB	P.AAB	: 1012
		7E 7C 0001C	CLRQ	-(SP)	: 1013
		7E D4 0001E	CLRL	-(SP)	: 1014
00000000G	00	07 FB 00020	CALLS	#7, SYSSGETJPI	: 1015
63	00000000'	EF 9E 00027	MOVAB	USER_LABEL, USER_VALUE+4	: 1016
		27 11 0002E	BRB	3\$	: 1017
0C	04	A2 B1 00030 1\$:	CMPW	4(R2), #12	: 1018
		18 1B 00034	BLEQU	2\$	: 1019
	00000000*	8F DD 00036	PUSHL	#<<<SETS_FACILITY@16>+4584>+2>	: 1020
	04	A2 9F 0003C	PUSHAB	4(R2)	: 1021
		01 DD 0003F	PUSHL	#1	: 1022
	00000000*	8F DD 00041	PUSHL	#<<<SETS_FACILITY@16>+4344>+2>	: 1023
00000000G	00	04 FB 00047	CALLS	#4, LIB\$STOP	: 1024
FC	A3 04	A2 3C 0004E 2\$:	MOVZWL	4(R2), USER_VALUE	: 1025
	63 08	A2 D0 00053	MOVL	8(R2), USER_VALUE+4	: 1026
	50	01 D0 00057 3\$:	MOVL	#1, R0	: 1027
		04 0005A	RET		: 1028
					: 1029
					: 1030

; Routine Size: 91 bytes. Routine Base: \$CODE\$ + 04DA

```

: 1060      1031 1 GLOBAL ROUTINE vprot_act (option_block, callback) =
: 1061      1032 1 +-
: 1062      1033 1 |
: 1063      1034 1 | This is the action routine for the PROTECTION qualifier of SET VOLUME.
: 1064      1035 1 | The protection is parsed and stored.
: 1065      1036 1 |
: 1066      1037 1 |--
: 1067      1038 2 BEGIN
: 1068      1039 2 |
: 1069      1040 2 LOCAL
: 1070      1041 2 |     status,           ! Status return
: 1071      1042 2 |     temp;             ! Temporary place for FPROT_VALUE
: 1072      1043 2 |
: 1073      1044 2 MAP option_block : REF BBLOCK; ! Define CLI block
: 1074      1045 2 |
: 1075      1046 2 |
: 1076      1047 2 |     Stuff the TPARSE block with the string
: 1077      1048 2 |
: 1078      1049 2 tparse_block[tpa$l_stringcnt] = .option_block[cli$w_qdvalsiz];
: 1079      1050 2 tparse_block[tpa$l_stringptr] = .option_block[cli$a_qdvaladr];
: 1080      1051 2 |
: 1081      1052 2 temp = .fprot_value;           ! Save contents of FPROT
: 1082      1053 2 fprot_value = 0;             ! Initialize file protection value
: 1083      1054 2 |
: 1084      1055 2 |
: 1085      1056 2 |     Now to parse the protection given. When finished, FPROT_VALUE will
: 1086      1057 2 |     have the following values:
: 1087      1058 2 |
: 1088      1059 2 |     FPROT_VALUE[low_word] = protection value
: 1089      1060 2 |     FPROT_VALUE[high_word] = group mask i.e. SYSTEM, OWNER, GROUP, WORLD
: 1090      1061 2 |
: 1091      1062 2 IF NOT (status = LIB$TPARSE(tparse_block,
: 1092      1063 2 |                               pro_state,
: 1093      1064 2 |                               pro_keys))
: 1094      1065 2 THEN SIGNAL_STOP(set$_facility^T6 + shr$_syntax + sts$_error,
: 1095      1066 2 |                               1,
: 1096      1067 2 |                               option_block[cli$q_qdvaldesc],
: 1097      1068 2 |                               .status);
: 1098      1069 2 |
: 1099      1070 2 vprot_value = .fprot_value;     ! Store VPROT value
: 1100      1071 2 fprot_value = .temp;           ! Restore FPROT value
: 1101      1072 2 |
: 1102      1073 2 RETURN true;
: 1103      1074 2 END;

```

```

                                003C 00000          .ENTRY VPROT_ACT, Save R2,R3,R4,R5          : 1031
                                EF 9E 00002        MOVAB TPARSE_BLOCK+8, R5
                                55 00000000' EF 9E 00002        MOVAB FPROT_VALUE, R4
                                54 00000000G 00 9E 00009        MOVAB OPTION_BLOCK, R2          : 1049
                                52          04 AC D0 00010        MOVZWL 4(R2), TPARSE_BLOCK+8
                                65          04 A2 3C 00014        MOVZWL 8(R2), TPARSE_BLOCK+12
                                04 A5          08 A2 D0 00018        MOVZWL 8(R2), TPARSE_BLOCK+12
                                53          08 A2 D0 00018        MOVZWL 8(R2), TPARSE_BLOCK+12
                                64 D0 0001D        MOVZWL 8(R2), TPARSE_BLOCK+12
                                64 D4 00020        MOVZWL 8(R2), TPARSE_BLOCK+12
                                CLRL FPROT_VALUE          : 1052
                                : 1053

```

SETACT  
V04-000

C 6  
16-Sep-1984 01:06:01  
14-Sep-1984 12:08:59

VAX-11 Bliss-32 V4.0-742  
[CLIUTL.SRC]SETACT.B32;1

Page 59  
(30)

		00000000'	EF	9F	00022	PUSHAB	PRO_KEYS	:	1062
		00000000'	EF	9F	00028	PUSHAB	PRO_STATE	:	
		F8	A5	9F	0002E	PUSHAB	TPARSE_BLOCK	:	
00000000G	00		03	FB	00031	CALLS	#3, LIB\$TPARSE	:	
	14		50	E8	00038	BLBS	STATUS, 1\$	:	
			50	DD	0003B	PUSHL	STATUS	:	1068
		04	A2	9F	0003D	PUSHAB	4(R2)	:	1067
			01	DD	00040	PUSHL	#1	:	
		00000000*	8F	DD	00042	PUSHL	#<<<SETS FACILITY@16>+4344>+2>	:	
00000000G	00		04	FB	00048	CALLS	#4, LIB\$STOP	:	
00000000G	00		64	DO	0004F	1\$:	MOVL	FPROT_VALUE, VPROT_VALUE	1070
	64		53	DO	00056	MOVL	TEMP, FPROT_VALUE	:	1071
	50		01	DO	00059	MOVL	#1, R0	:	1073
			04	0005C	RET			:	1074

; Routine Size: 93 bytes. Routine Base: \$CODE\$ + 0535

```

1105 1075 1 GLOBAL ROUTINE vrsn_act (option_block,callback) =
1106 1076 1 |++
1107 1077 1 |
1108 1078 1 | This is the action routine for the VERSION_LIMIT qualifier. The value of
1109 1079 1 | the version limit is collected.
1110 1080 1 |
1111 1081 1 |--
1112 1082 2 BEGIN
1113 1083 2
1114 1084 2 LOCAL
1115 1085 2     status,
1116 1086 2     desc : BBLOCK[dsc$c_s_bln];
1117 1087 2
1118 1088 2 MAP
1119 1089 2     option_block : REF BBLOCK;           ! Define the CLI options block
1120 1090 2
1121 1091 2     vrsn_value = 32767;                 ! Preset to no limit
1122 1092 2
1123 1093 2 |
1124 1094 2 | See if a value was present. If yes, use it. Otherwise, use default
1125 1095 2 |
1126 1096 2 IF .option_block[cli$q_dvalsiz] EQL 0
1127 1097 2 THEN RETURN true;
1128 1098 2
1129 1099 2 IF NOT (status = LIB$CVT_DTB(.option_block[cli$q_dvalsiz],
1130 1100 2     .option_block[cli$a_qdvaladr],
1131 1101 2     vrsn_value))
1132 1102 2 THEN
1133 1103 2     BEGIN
1134 1104 2     SIGNAL_STOP(set$_facility^16 + shr$_syntax + sts$k_error,
1135 1105 2         1,
1136 1106 2         option_block[cli$q_qdvaldesc],
1137 1107 2         .status);
1138 1108 2     END
1139 1109 2 ELSE
1140 1110 2     BEGIN
1141 1111 2     IF NOT (.vrsn_value GEQ 0 AND .vrsn_value LEQ 65535)
1142 1112 2     THEN SIGNAL_STOP(set$_facility^16 + shr$_syntax + sts$k_error,
1143 1113 2         1,
1144 1114 2         option_block[cli$q_qdvaldesc],
1145 1115 2         set$_facility^16 + shr$_valerr + sts$k_error);
1146 1116 2     END;
1147 1117 2
1148 1118 2 RETURN true;
1149 1119 1 END;

```

			000C 0000	.ENTRY	VRSN_ACT, Save R2,R3	: 1075
53	00000000G	00	9E 00002	MOVAB	VRSN_VALUE, R3	...
5E		08	C2 00009	SUBL2	#8, SP	...
63	7FFF	8F	3C 0000C	MOVZWL	#32767, VRSN_VALUE	: 1091
52	04	AC	D0 00011	MOVL	OPTION_BLOCK, R2	: 1096
	04	A2	B5 00015	TSTW	4(R2)	...
		3D	13 00018	BEQL	4\$	...

SETACT  
V04-000

E 6  
16-Sep-1984 01:06:01 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:08:59 [CLIUTL.SRC]SETACT.B32;1

Page 61  
(31)

			53	DD	0001A		PUSHL	R3	:	1099
		08	A2	DD	0001C		PUSHL	8(R2)	:	1100
		04	A2	3C	0001F		MOVZWL	4(R2), -(SP)	:	1099
00000000G	7E		03	FB	00023		CALLS	#3, LIB\$CVT_DTB	:	
	00		50	EB	0002A		BLBS	STATUS, 1\$	:	
	04		50	DD	0002D		PUSHL	STATUS	:	1107
			14	11	0002F		BRB	3\$	:	1106
	50		63	D0	00031	1\$:	MOVL	VRSN_VALUE, R0	:	1111
			09	19	00034		BLSS	2\$	:	
0000FFFF	8F		50	D1	00036		C MPL	R0, #65535	:	
			18	15	0003D		BLEQ	4\$	:	
		00000000*	8F	DD	0003F	2\$:	PUSHL	#<<<SETS_FACILITY@16>+4584>+2>	:	1115
		04	A2	9F	00045	3\$:	PUSHAB	4(R2)	:	1114
			01	DD	00048		PUSHL	#1	:	
		00000000*	8F	DD	0004A		PUSHL	#<<<SETS_FACILITY@16>+4344>+2>	:	
00000000G	00		04	FB	00050		CALLS	#4, LIB\$STOP	:	
	50		01	D0	00057	4\$:	MOVL	#1, R0	:	1118
			04	0005A			RET		:	1119

: Routine Size: 91 bytes, Routine Base: \$CODE\$ + 0592

```

: 1151      1120 1 GLOBAL ROUTINE window_act (option_block, callback) =
: 1152      1121 1 !++
: 1153      1122 1 !
: 1154      1123 1 ! This is the action routine for the /WINDOWS qualifier. It retrieves the
: 1155      1124 1 ! value and performs bounds checking on it.
: 1156      1125 1 !
: 1157      1126 1 !--
: 1158      1127 2 BEGIN
: 1159      1128 2
: 1160      1129 2 LOCAL
: 1161      1130 2     status,           ! Status return
: 1162      1131 2     desc : BBLOCK[dsc$c_s_bln]; ! General descriptor
: 1163      1132 2
: 1164      1133 2 MAP option_block : REF BBLOCK; ! Define the CLI block
: 1165      1134 2
: 1166      1135 2 window_value = 7;           ! Set up the default
: 1167      1136 2
: 1168      1137 2 !
: 1169      1138 2 ! If a value was specified, use it; otherwise, use the default.
: 1170      1139 2
: 1171      1140 2 IF .option_block[cli$w_qdvalsiz] EQL 0
: 1172      1141 2 THEN RETURN true;
: 1173      1142 2
: 1174      1143 2 !
: 1175      1144 2 ! Convert the value
: 1176      1145 2
: 1177      1146 2 IF NOT (status = LIB$CVT_DTB(.option_block[cli$w_qdvalsiz],
: 1178      1147 2     .option_block[cli$a_qdvaladr],
: 1179      1148 2     window_value))
: 1180      1149 2 THEN SIGNAL_STOP(set$_facility^16 + shr$_syntax + sts$k_error, ! Signal a syntax error
: 1181      1150 2     1,
: 1182      1151 2     option_block[cli$q_qdvaldesc],
: 1183      1152 2     .status)
: 1184      1153 2 ELSE
: 1185      1154 2 BEGIN
: 1186      1155 2     IF NOT (.window_value GEQ 7           ! Check that value is in range
: 1187      1156 2     AND
: 1188      1157 2     .window_value LEQ 80)
: 1189      1158 2     THEN SIGNAL_STOP(set$_facility^16 + shr$_syntax + sts$k_error, ! If not, exit with an error.
: 1190      1159 2     1,
: 1191      1160 2     option_block[cli$q_qdvaldesc],
: 1192      1161 2     set$_facility^16 + shr$_valerr + sts$k_error);
: 1193      1162 2 END;
: 1194      1163 2 RETURN true;
: 1195      1164 1 END;

```

		000C 0000	.ENTRY WINDOW_ACT, Save R2,R3	: 1120
53	00000000G	00 9E 00002	MOVAB WINDOW_VALUE, R3	...
5E		08 C2 00009	SUBL2 #8, SP	...
63		07 D0 0000C	MOVL #7, WINDOW_VALUE	: 1135
52	04	AC D0 0000F	MOVL OPTION_BLOCK, R2	: 1140
	04	A2 B5 00013	TSTW 4(R2)	...
		40 13 00016	BEQL 4\$	...



00000000G	7E 00 04	08 04	53 DD 00018 A2 DD 0001A A2 3C 0001D 03 FB 00021 50 EB 00028 50 DD 0002B 17 11 0002D 63 D0 0002F 1\$: 50 D1 00032 09 19 00035 00000050 8F 50 D1 00037 18 15 0003E 00000000* 8F DD 00040 2\$: 04 A2 9F 00046 3\$: 01 DD 00049 00000000* 8F DD 0004B 00000000G 00 04 FB 00051 50 01 D0 00058 4\$: 04 0005B	PUSHL R3 PUSHL 8(R2) MOVZWL 4(R2), -(SP) CALLS #3, LIB\$CVT_DTB BLBS STATUS, 1\$ PUSHL STATUS BRB 3\$ MOVL WINDOW_VALUE, R0 CML R0, #7 BLSS 2\$ CML R0, #80 BLEQ 4\$ PUSHL #<<<SETS_FACILITY@16>+4584>+2> PUSHAB 4(R2) PUSHL #1 PUSHL #<<<SETS_FACILITY@16>+4344>+2> CALLS #4, LIB\$STOP MOVL #1, R0 RET	: 1146 : 1147 : 1146 : 1152 : 1151 : 1155 : 1157 : 1161 : 1160 : 1163 : 1164
-----------	----------------	----------	---	--	--

: Routine Size: 92 bytes. Routine Base: \$CODE\$ + 05ED

SETACT  
V04-000

H 6  
16-Sep-1984 01:06:01  
14-Sep-1984 12:08:59

VAX-11 Bliss-32 V4.0-742  
[CLIUTL.SRC]SETACT.B32;1

: 1197 1165 1 END  
: 1198 1166 0 ELUDOM

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
SOWNS	56	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
_LIB\$KEYOS	40	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(1)
_LIB\$STATES	642	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(1)
_LIB\$KEYIS	104	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(1)
\$CODES	1609	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
ABS	0	NOVEC, NOWRT, NORD, NOEXE, NOSHR, LCL, ABS, CON, NOPIC, ALIGN(0)
\$PLITS	32	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	Symbols		Pages Mapped	Processing Time
	Total	Loaded Percent		
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	30 0	1000	00:01.8
_\$255\$DUA28:[SYSLIB]CLIMAC.L32;1	14	0 0	9	00:00.1
_\$255\$DUA28:[SYSLIB]TPAMAC.L32;1	42	29 69	14	00:00.2

COMMAND QUALIFIERS

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

: Size: 1609 code + 874 data bytes  
: Run Time: 01:06.7  
: Elapsed Time: 03:44.0  
: Lines/CPU Min: 1049  
: Lexemes/CPU-Min: 69494  
: Memory Used: 274 pages  
: Compilation Complete



0052 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

