

[illegible]

```
SSSSSSSS HH HH 000000 WW WW AAAAAA CCCCCCCC LL
SSSSSSSS HH HH 000000 WW WW AAAAAA CCCCCCCC LL
SS SS HH HH 00 00 WW WW AA AA CC LL
SS SS HH HH 00 00 WW WW AA AA CC LL
SS SS HH HH 00 00 WW WW AA AA CC LL
SSSSSS HH HH 00 00 WW WW AA AA CC LL
SSSSSS HH HH 00 00 WW WW AA AA CC LL
SS HH HH 00 00 WW WW AAAAAAAAAA CC LL
SS HH HH 00 00 WW WW AAAAAAAAAA CC LL
SS HH HH 00 00 WWW WWW AA AA CC LL
SSSSSS HH HH 000000 WW WW AA AA CCCCCCCC LL
SSSSSS HH HH 000000 WW WW AA AA CCCCCCCC LLLLLLLLLL
                                         ....
                                         ....
                                         ....
                                         ....
```

```
LL LL SSSSSSSS
LL LL SSSSSSSS
LL LL SS
LL LL SS
LL LL SS
LL LL SSSSSS
LL LL SSSSSS
LL LL SS
LL LL SS
LL LL SS
LLLLLLLLLL
LLLLLLLLLL

IIIIII
IIIIII
II
II
II
II
II
II
II
II
II
IIIIII
IIIIII

SSSSSSSS
SSSSSSSS
SS
SS
SS
SS
SSSSSS
SSSSSS
SS
SS
SS
SS
SSSSSSS
SSSSSSS
```

```
1 0001 0 MODULE SHOW$ACL (
2 0002 0     LANGUAGE (BLISS32),
3 0003 0     IDENT = 'V04-000',
4 0004 0     ADDRESSING_MODE (EXTERNAL = GENERAL)
5 0005 0 ) =
6 0006 1 BEGIN
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 *  ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 *  TRANSFERRED.
20 0020 1 *
21 0021 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 *  CORPORATION.
24 0024 1 *
25 0025 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 ++
32 0032 1
33 0033 1 FACILITY:      SHOW utility
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1     This module contains all the routines necessary to support the
38 0038 1     DCL command SHOW ACL.
39 0039 1
40 0040 1 ENVIRONMENT:
41 0041 1
42 0042 1     VAX/VMS operating system, user mode utilities.
43 0043 1
44 0044 1 --
45 0045 1
46 0046 1
47 0047 1 AUTHOR:      L. Mark Pilant      CREATION DATE: 17-Jan-1984
48 0048 1
49 0049 1 MODIFIED BY:
50 0050 1
51 0051 1     V03-003 LMP0230      L. Mark Pilant,      16-Apr-1984 10:53
52 0052 1     Track interface changes to $CHANGE_ACL system service.
53 0053 1
54 0054 1     V03-002 LMP0223      L. Mark Pilant,      6-Apr-1984 12:53
55 0055 1     Use the correct amount of storage for the lock block.
56 0056 1
57 0057 1     V03-001 LMP0213      L. Mark Pilant,      24-Mar-1984 12:23
```

SHOW\$ACL
V04-000

J 3
16-Sep-1984 00:07:24 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 11:52:36 [ACLEDIT.SRC]SHOWACL.B32;1

Page 2
(1)

:	58	0058	1	:	Add support for locking and unlocking the object's ACL. Also, modify it so that the DCL commands SET ACL and SHOW ACL call the same image.
:	59	0059	1	:	
:	60	0060	1	:	
:	61	0061	1	:	
:	62	0062	1	!*	
:	63	0063	1	:	
:	64	0064	1	LIBRARY 'SYSS\$LIBRARY:LIB';	

```

: 66      0065 1 ! Routines contained within this module.
: 67      0066 1
: 68      0067 1 FORWARD ROUTINE
: 69      0068 1     SHOW_ACL;                                ! Main processing routine
: 70      0069 1
: 71      0070 1 ! Define common error message codes.
: 72      0071 1
: 73      P 0072 1 $SHR_MSGDEF      (SHOW, 120, LOCAL,
: 74      P 0073 1     (READERR, SEVERE)
: 75      0074 1     );
: 76      0075 1
: 77      0076 1 ! Define necessary macros.
: 78      0077 1
: 79      M 0078 1 MACRO    SIGNAL    (ARG) =
: 80      M 0079 1     BEGIN
: 81      M 0080 1     EXTERNAL ROUTINE      LIB$SIGNAL;
: 82      M 0081 1     LIB$SIGNAL (ARG %IF %LENGTH-1 GTR 0 %THEN, %REMAINING %FI);
: 83      M 0082 1     END
: 84      0083 1     %;
: 85      0084 1
: 86      0085 1 ! External routine definitions.
: 87      0086 1
: 88      0087 1 EXTERNAL ROUTINE
: 89      0088 1     CLISGET_VALUE,                ! Get qualifier value
: 90      0089 1     CLISPRESENT,                  ! See if qualifier present
: 91      0090 1     LIB$PUT_OUTPUT;               ! General output routine

```

```
0091 1 GLOBAL RCUTINE SHOW_ACL =
0092 1
0093 1 !++
0094 1
0095 1 FUNCTIONAL DESCRIPTION:
0096 1
0097 1 This routine is the main routine. It parses the command line to
0098 1 determine the name and type of the object whose ACL is to be
0099 1 displayed.
0100 1
0101 1 !--
0102 1
0103 2 BEGIN
0104 2
0105 2 LOCAL
0106 2 ACL_LOCKID : $BBLOCK [ACL$$_RLOCK_ACL], ! Lock-id for ACL lock
0107 2 OBJECT_TYPE, : ! Object's type code
0108 2 OBJECT_NAME : $BBLOCK [DSC$$_S_BLN], ! Object name descr
0109 2 OBJECT_FAB : $FAB_DECL, ! Object FAB
0110 2 OBJECT_NAM : $NAM_DECL, ! Object NAME block
0111 2 OBJECT_EXP_NAME : $BBLOCK [NAM$$_MAXRSS], ! Expanded name string
0112 2 OBJECT_RES_NAME : $BBLOCK [NAM$$_MAXRSS], ! Resultant name string
0113 2 OBJECT_CHAN, : ! Channel for object
0114 2 GETDVI_ARGS : BLOCKVECTOR [3, ITM$$_ITEM, BYTE], ! $GETDVI item list
0115 2 DEVICE_CLASS, : ! Output device class
0116 2 DISPLAY_WIDTH, : ! Output device width
0117 2
0118 2 ACE_BIN_DESC : $BBLOCK [DSC$$_S_BLN], ! Binary ACE descr
0119 2 ACE_STORAGE : $BBLOCK [ACL$$_READACE] VOLATILE, ! Binary ACE storage
0120 2 ACE_TXT_DESC : $BBLOCK [DSC$$_S_BLN], ! Text ACE descr
0121 2 ACE_TEXT : VECTOR [512, BYTE], ! Text ACE storage
0122 2 ATR_ARGLIST : BLOCKVECTOR [2, ITM$$_ITEM, BYTE], ! $CHANGE_ACL item list
0123 2 ACL_CONTEXT, : ! $CHANGE_ACL context
0124 2 OUTPUT_DESC : $BBLOCK [DSC$$_S_BLN], ! Output line descr
0125 2 OUTPUT_TEXT : VECTOR [512, BYTE], ! Output line storage
0126 2 IO_STATUS : VECTOR [4, WORD], ! I/O status block
0127 2 STATUS; : ! Routine return status
0128 2
0129 2
0130 2 EXTERNAL LITERAL
0131 2 SHOW$_OBJLOCKED; ! object locked error message
0132 2
0133 2 BIND
0134 2 OBJ_TYPE_NAMES = UPLIT (
0135 2 0,
0136 2 $DESCRIPTOR ('file'),
0137 2 $DESCRIPTOR ('device'),
0138 2 $DESCRIPTOR ('print/batch queue'),
0139 2 $DESCRIPTOR ('event cluster'),
0140 2 $DESCRIPTOR ('logical name table'),
0141 2 $DESCRIPTOR ('process'),
0142 2 $DESCRIPTOR ('global section')
0143 2 ) : VECTOR;
0144 2
0145 2 ! Initialize all necessary storage.
0146 2
0147 2 CH$FILL (0, DSC$$_S_BLN, OBJECT_NAME);
0148 2 CH$FILL (0, DSC$$_S_BLN, ACE_BIN_DESC);
0149 2 CH$FILL (0, DSC$$_S_BLN, ACE_TXT_DESC);
```

```
150 0148 2 CH$FILL (0, DSC$S_BLN, OUTPUT_DESC);
151 0149 2 CH$FILL (0, 3*ITM$S_ITEM, GETDVT_ARGS);
152 0150 2 CH$FILL (0, 2*ITM$S_ITEM, ATR_ARGLIST);
153 0151 2 CH$FILL (0, 4*2, IO_STATUS);
154 0152 2
155 0153 2 OBJECT_NAME[DSC$B_CLASS] = DSC$K_CLASS_D;
156 0154 2 OBJECT_CHAN = 0;
157 0155 2 ACE_BIN_DESC[DSC$A_POINTER] = ACE_STORAGE;
158 0156 2 ACE_TXT_DESC[DSC$A_POINTER] = ACE_TEXT;
159 0157 2 OUTPUT_DESC[DSC$A_POINTER] = OUTPUT_TEXT;
160 0158 2 OUTPUT_TEXT[0] = 0;
161 0159 2
162 0160 2 ACL_CONTEXT = 0;
163 0161 2
164 0162 2 ! Get the type code and name of the desired object.
165 0163 2
166 0164 2 OBJECT_TYPE = ACL$C_FILE;
167 0165 2 IF CL$PRESENT ($DESCRIPTOR ('OBJECT_TYPE.FILE')) THEN OBJECT_TYPE = ACL$C_FILE;
168 0166 2 IF CL$PRESENT ($DESCRIPTOR ('OBJECT_TYPE.DEVICE')) THEN OBJECT_TYPE = ACL$C_DEVICE;
169 0167 2 IF CL$PRESENT ($DESCRIPTOR ('OBJECT_TYPE.QUEUE')) THEN OBJECT_TYPE = ACL$C_JOBCTL_QUEUE;
170 0168 2 IF CL$PRESENT ($DESCRIPTOR ('OBJECT_TYPE.EVENT_CLUSTER')) THEN OBJECT_TYPE = ACL$C_COMMON_EF_CLUSTER;
171 0169 2 IF CL$PRESENT ($DESCRIPTOR ('OBJECT_TYPE.LOGICAL_NAME_TABLE')) THEN OBJECT_TYPE = ACL$C_LOGICAL_NAME_TABLE;
172 0170 2 IF CL$PRESENT ($DESCRIPTOR ('OBJECT_TYPE.PROCESS')) THEN OBJECT_TYPE = ACL$C_PROCESS;
173 0171 2 IF CL$PRESENT ($DESCRIPTOR ('OBJECT_TYPE.GLOBAL_SECTION')) THEN OBJECT_TYPE = ACL$C_GLOBAL_SECTION;
174 0172 2
175 0173 2 CL$GET_VALUE ($DESCRIPTOR ('INPUT'), OBJECT_NAME);
176 0174 2
177 0175 2 ! Attempt to obtain a read lock for the object.
178 0176 2
179 0177 2 ATR_ARGLIST[0, ITM$W_ITMCD] = ACL$C_RLOCK_ACL;
180 0178 2 ATR_ARGLIST[0, ITM$W_BUFSIZ] = ACL$S_RLOCK_ACL;
181 0179 2 ATR_ARGLIST[0, ITM$L_BUFADR] = ACL_LOCKID;
182 P 0180 2 STATUS = $CHANGE_ACL (CHAN = .OBJECT_CHAN,
183 P 0181 2 OBJTYP = OBJECT_TYPE,
184 P 0182 2 OBJNAM = OBJECT_NAME,
185 0183 2 ITMLST = ATR_ARGLIST);
186 0184 2 IF NOT .STATUS
187 0185 2 THEN
188 0186 2 BEGIN
189 0187 2 IF .STATUS EQL SSS$_NOTQUEUED THEN STATUS = SHOW$_OBJLOCKED;
190 0188 2 SIGNAL (.STATUS);
191 0189 2 RETURN .STATUS OR STS$M_INHIB_MSG;
192 0190 2 END;
193 0191 2
194 0192 2 ! If the object is a file, open it to improve performance.
195 0193 2
196 0194 2 IF .OBJECT_TYPE EQL ACL$C_FILE
197 0195 2 THEN
198 0196 2 BEGIN
199 P 0197 2 $FAB_INIT (FAB = OBJECT_FAB,
200 P 0198 2 FAC = GET,
201 P 0199 2 FNA = .OBJECT_NAME[DSC$A_POINTER],
202 P 0200 2 FNS = .OBJECT_NAME[DSC$W_LENGTH],
203 P 0201 2 FOP = UFO,
204 P 0202 2 NAM = OBJECT_NAM,
205 0203 2 SHR = <GET, PUT, UPI>);
206 P 0204 2 $NAM_INIT (NAM = OBJECT_NAM,
```

```

: 207 P 0205 3          ESA = OBJECT_EXP_NAME,
: 208 PP 0206 3          ESS = NAM$C_MAXRSS,
: 209 P 0207 3          RSA = OBJECT_RES_NAME,
: 210 0208 3          RSS = NAM$C_MAXRSS);
: 211 0209 3          STATUS = $OPEN (FAB = OBJECT_FAB);
: 212 0210 3
: 213 0211 3      ! Get the actual name of the file, if possible.
: 214 0212 3
: 215 0213 3      IF .OBJECT_NAM[NAM$B_RSL] NEQ 0
: 216 0214 3      THEN
: 217 0215 4          BEGIN
: 218 0216 4              OBJECT_NAME[DSC$W_LENGTH] = .OBJECT_NAM[NAM$B_RSL];
: 219 0217 4              OBJECT_NAME[DSC$A_POINTER] = .OBJECT_NAM[NAM$[_RSA]];
: 220 0218 4          END
: 221 0219 3      ELSE IF .OBJECT_NAM[NAM$B_ESL] NEQ 0
: 222 0220 3      THEN
: 223 0221 4          BEGIN
: 224 0222 4              OBJECT_NAME[DSC$W_LENGTH] = .OBJECT_NAM[NAM$B_ESL];
: 225 0223 4              OBJECT_NAME[DSC$A_POINTER] = .OBJECT_NAM[NAM$[_ESA]];
: 226 0224 4          END
: 227 0225 3      ELSE
: 228 0226 4          BEGIN
: 229 0227 4              OBJECT_NAME[DSC$W_LENGTH] = .OBJECT_FAB[FAB$B_FNS];
: 230 0228 4              OBJECT_NAME[DSC$A_POINTER] = .OBJECT_FAB[FAB$[_FNA]];
: 231 0229 4          END;
: 232 0230 3
: 233 0231 3      ! If any errors occurred, abort now.
: 234 0232 3
: 235 0233 3      IF NOT .STATUS
: 236 0234 3      THEN
: 237 0235 4          BEGIN
: 238 P 0236 4              SIGNAL (SHOW$_READERR, 1, OBJECT_NAME, .OBJECT_FAB[FAB$L_STS],
: 239 0237 4                  .OBJECT_FAB[FAB$L_STV]);
: 240 0238 4              RETURN .STATUS OR STS$M_INHIB_MSG;
: 241 0239 4          END;
: 242 0240 3      OBJECT_CHAN = .OBJECT_FAB[FAB$L_STV];
: 243 0241 3      END;
: 244 0242 2
: 245 0243 2      ! Get the width of SYS$OUTPUT, the display device.
: 246 0244 2
: 247 0245 2      GETDVI_ARGS[0, ITMSW_ITMCD] = DVI$_DEVCLASS;
: 248 0246 2      GETDVI_ARGS[0, ITMSW_BUFSIZ] = 4;
: 249 0247 2      GETDVI_ARGS[0, ITMSL_BUFADR] = DEVICE_CLASS;
: 250 0248 2      GETDVI_ARGS[1, ITMSW_ITMCD] = DVI$_DEVBUFSIZ;
: 251 0249 2      GETDVI_ARGS[1, ITMSW_BUFSIZ] = 4;
: 252 0250 2      GETDVI_ARGS[1, ITMSL_BUFADR] = DISPLAY_WIDTH;
: 253 0251 2
: 254 P 0252 2      STATUS = $GETDVI (DEVNAM = $DESCRIPTOR ('SYS$OUTPUT'),
: 255 P 0253 2          ITMLST = GETDVI_ARGS,
: 256 0254 2          IOSB = IO_STATUS);
: 257 0255 2      IF .STATUS THEN STATUS = .IO_STATUS[0];
: 258 0256 2      IF NOT .STATUS
: 259 0257 2      THEN
: 260 0258 3          BEGIN
: 261 0259 3              SIGNAL (.STATUS);
: 262 0260 3              RETURN .STATUS OR STS$M_INHIB_MSG;
: 263 0261 2          END;
```

```
264 0262 2 IF .DEVICE_CLASS NEQ DC$_TERM THEN DISPLAY_WIDTH = 132;
265 0263 2
266 0264 2 ! Read, format, and display all the ACEs in the object's ACL until the end
267 0265 2 ! of the ACL is seen. Or until an error occurs.
268 0266 2
269 0267 2 WHILE 1
270 0268 2 DO
271 0269 2 BEGIN
272 0270 2 ATR_ARGLIST[0, ITMSW_ITMCO] = ACL$_READACE;
273 0271 2 ATR_ARGLIST[0, ITMSW_BUFSIZ] = ACL$_READACE;
274 0272 2 ATR_ARGLIST[0, ITMSL_BUFADR] = ACE_STORAGE;
275 P 0273 2 STATUS = $CHANGE_ACL (CHAN = .OBJECT_CHAN,
276 P 0274 2 OBJTYP = OBJECT_TYPE,
277 P 0275 2 OBJNAM = OBJECT_NAME,
278 P 0276 2 ITMLST = ATR_ARGLIST,
279 0277 2 CONTXT = ACL_CONTEXT);
280 0278 2 IF NOT .STATUS
281 0279 2 THEN
282 0280 2 BEGIN
283 0281 2 IF .STATUS EQL SS$_ACLEMPY OR .STATUS EQL SS$_NOMOREACE THEN EXITLOOP;
284 0282 2 SIGNAL (SHOW$_READERR, 1, OBJECT_NAME, .STATUS);
285 0283 2 RETURN SHOW$_READERR OR STSM_INHIB_MSG;
286 0284 2 END;
287 0285 2
288 0286 2 ! If the ACE is tagged as being hidden, don't display it.
289 0287 2
290 0288 2 IF NOT .ACE_STORAGE[ACE$_V_HIDDEN]
291 0289 2 THEN
292 0290 2 BEGIN
293 0291 2
294 0292 2 ! Type out the banner line indicating the object type, name, and the current
295 0293 2 ! date and time if this is the first ACE.
296 0294 2
297 0295 2 IF .OUTPUT_TEXT[0] EQL 0
298 0296 2 THEN
299 0297 2 BEGIN
300 0298 2 OUTPUT_DESC[DSC$_LENGTH] = 512;
301 P 0299 2 STATUS = $FAO ($DESCRIPTOR ('Object type: !AS, Object name: !AS, on !%D!/''),
302 P 0300 2 OUTPUT_DESC,
303 P 0301 2 OUTPUT_DESC,
304 P 0302 2 .OBJ_TYPE_NAMES[.OBJECT_TYPE],
305 P 0303 2 OBJECT_NAME,
306 0304 2 0);
307 0305 2 LIB$PUT_OUTPUT (OUTPUT_DESC);
308 0306 2 END;
309 0307 2
310 0308 2 ! Format and display the ACE.
311 0309 2
312 0310 2 ACE_BIN_DESC[DSC$_LENGTH] = .ACE_STORAGE[ACE$_B_SIZE];
313 0311 2 ACE_TXT_DESC[DSC$_LENGTH] = 512;
314 P 0312 2 STATUS = $FORMAT_ACL (ACLEN = ACE_BIN_DESC,
315 P 0313 2 ACLEN = ACE_TXT_DESC[DSC$_LENGTH],
316 P 0314 2 ACLSTR = ACE_TXT_DESC,
317 P 0315 2 WIDTH = DISPLAY_WIDTH,
318 0316 2 TRMDSC = $DESCRIPTOR (%CHAR (13), %CHAR (10)),
319 0317 2 INDENT = %REF (10));
320 0318 2 IF NOT .STATUS
```

```
321      0319 4      THEN
322      0320 5      BEGIN
323      0321 5      SIGNAL (.STATUS);
324      0322 5      RETURN .STATUS OR ST$M_INHIB_MSG;
325      0323 4      END;
326      0324 4      LIB$PUT_OUTPUT (ACE_TXT_DESC);
327      0325 3      END;
328      0326 3      END;
329      0327 3
330      0328 2      ! Check for an empty ACL.
331      0329 2
332      0330 2      IF .STATUS EQL S$$_ACEMPTY
333      0331 2      THEN RETURN S$$_ACEMPTY
334      0332 2      ELSE RETURN 1;
335      0333 2
336      0334 1      END;
INFO#250      L1:0262
: Referenced LOCAL symbol DEVICE_CLASS is probably not initialized
```

! End of routine SHOW_ACL

```
      .TITLE  SHOW$ACL
      .IDENT  \V04-000\
      .PSECT  $SPLIT$,NOWRT,NOEXE,2
      65 6C 69 66 00000 P.AAC: .ASCII \file\
      00000004 00004 P.AAB: .LONG 4
      00000000 00008 P.AAB: .ADDRESS P.AAC
      65 63 69 76 65 64 0000C P.AAE: .ASCII \device\
      00012 P.AAE: .BLKB 2
      00000006 00014 P.AAD: .LONG 6
      00000000 00018 P.AAD: .ADDRESS P.AAE
      65 75 71 20 68 63 74 61 62 2F 74 6E 69 72 70 0001C P.AAG: .ASCII \print/batch queue\
      65 75 0002B P.AAG: .BLKB 3
      0002D P.AAF: .LONG 17
      00000011 00030 P.AAF: .ADDRESS P.AAG
      00000000 00034 P.AAI: .ASCII \event cluster\
      72 65 74 73 75 6C 63 20 74 6E 65 76 65 00038 P.AAI: .BLKB 3
      00045 P.AAH: .LONG 13
      00000000 0004C P.AAH: .ADDRESS P.AAI
      61 74 20 65 6D 61 6E 20 6C 61 63 69 67 6F 6C 00050 P.AAK: .ASCII \logical name table\
      65 6C 62 0005F P.AAK: .BLKB 2
      00062 P.AAJ: .LONG 18
      00000012 00064 P.AAJ: .ADDRESS P.AAK
      00000000 00068 P.AAM: .ASCII \process\
      73 73 65 63 6F 72 70 0006C P.AAM: .BLKB 1
      00073 P.AAL: .LONG 7
      00000007 00074 P.AAL: .ADDRESS P.AAM
      00000000 00078 P.AAO: .ASCII \global section\
      6E 6F 69 74 63 65 73 20 6C 61 62 6F 6C 67 0007C P.AAO: .BLKB 2
      0008A P.AAN: .LONG 14
      0000000E 0008C P.AAN: .ADDRESS P.AAO
      00000000 00090 P.AAA: .LONG 0
      00000000 00094 P.AAA: .ADDRESS P.AAB, P.AAD, P.AAF, P.AAH, P.AAJ, -
      00000000 00098 P.AAA: P.AAL, P.AAN
      00000000 000B0
```

SHOW\$ACL
V04-000

D 4
16-Sep-1984 00:07:24 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 11:52:36 [ACLEDT.SRC]SHOWACL.B32;1

Page 9
(3)

4C	49	46	2E	45	50	59	54	5F	54	43	45	4A	42	4F	000B4	P.AAQ:	.ASCII	\OBJECT_TYPE.FILE\	:
														45	000C3				:
														00000010	000C4	P.AAP:	.LONG	16	:
56	45	44	2E	45	50	59	54	5F	54	43	45	4A	42	4F	000C8		.ADDRESS	P.AAQ	:
												45	43	49	000CC	P.AAS:	.ASCII	\OBJECT_TYPE.DEVICE\	:
															000DB				:
														00000012	000DE		.BLKB	2	:
45	55	51	2E	45	50	59	54	5F	54	43	45	4A	42	4F	000E0	P.AAR:	.LONG	18	:
														00000000	000E4		.ADDRESS	P.AAS	:
												45	55		000E8	P.AAU:	.ASCII	\OBJECT_TYPE.QUEUE\	:
															000F7				:
														00000011	000F9		.BLKB	3	:
45	56	45	2E	45	50	59	54	5F	54	43	45	4A	42	4F	000FC	P.AAT:	.LONG	17	:
														00000000	00100		.ADDRESS	P.AAU	:
					52	45	54	53	55	4C	43	5F	54	4E	00104	P.AAW:	.ASCII	\OBJECT_TYPE.EVENT_CLUSTER\	:
															00113				:
														00000019	0011D		.BLKB	3	:
47	4F	4C	2E	45	50	59	54	5F	54	43	45	4A	42	4F	00120	P.AAV:	.LONG	25	:
45	4C	42	41	54	5F	45	4D	41	4E	5F	4C	41	43	49	00124		.ADDRESS	P.AAW	:
															00128	P.AAY:	.ASCII	\OBJECT_TYPE.LOGICAL_NAME_TABLE\	:
															00137				:
														0000001E	00146		.BLKB	2	:
4F	52	50	2E	45	50	59	54	5F	54	43	45	4A	42	4F	00148	P.AAX:	.LONG	30	:
												53	53	45	0014C		.ADDRESS	P.AAY	:
															00150	P.ABA:	.ASCII	\OBJECT_TYPE.PROCESS\	:
															0015F				:
														00000013	00163		.BLKB	1	:
4F	4C	47	2E	45	50	59	54	5F	54	43	45	4A	42	4F	00164	P.AAZ:	.LONG	19	:
				4E	4F	49	54	43	45	53	5F	4C	41	42	00168		.ADDRESS	P.ABA	:
															0016C	P.ABC:	.ASCII	\OBJECT_TYPE.GLOBAL_SECTION\	:
															0017B				:
														0000001A	00186		.BLKB	2	:
														00000000	00188	P.ABB:	.LONG	26	:
															0018C		.ADDRESS	P.ABC	:
														54	55	50	.ASCII	\INPUT\	:
															00195	P.ABE:			:
														00000005	00198	P.ABD:	.LONG	5	:
														00000000	0019C		.ADDRESS	P.ABE	:
					54	55	50	54	55	4F	24	53	59	53	001A0	P.ABG:	.ASCII	\SYS\$OUTPUT\	:
															001AA		.BLKB	2	:
														0000000A	001AC	P.ABF:	.LONG	10	:
														00000000	001B0		.ADDRESS	P.ABG	:
41	21	20	3A	65	70	79	74	20	74	63	65	6A	62	4F	001B4	P.ABI:	.ASCII	\Object type: !AS, Object name: !AS, on\	:
65	6D	61	6E	20	74	63	65	6A	62	4F	20	20	2C	53	001C3				:
					6E	6F	20	20	2C	53	41	21	20	3A	001D2				:
									2F	21	44	25	21	20	001DC		.ASCII	\ !%D!/\	:
															001E2		.BLKB	2	:
														0000002E	001E4	P.ABH:	.LONG	46	:
														00000000	001E8		.ADDRESS	P.ABI	:
														0D	001EC	P.ABK:	.ASCII	<13>	:
														0A	001ED		.ASCII	<10>	:
															001EE		.BLKB	2	:
														00000002	001F0	P.ABJ:	.LONG	2	:
														00000000	001F4		.ADDRESS	P.ABK	:

OBJ_TYPE_NAMES= P.AAA
EXTRN CLISGET_VALUE, CLISPRESENT

08 00
08 00
08 00
08 00
24 00
18 00
08 00

OFFC 00000
5B 00000000G 00 9E 00002
5A 00000000G 00 9E 00009
59 0000' CF 9E 00010
58 00000000G 00 9E 00015
5E F7D4 CE 9E 0001C
6E 00 2C 00021
F8 AD 00026
6E 00 2C 00028
FD1C CD 0002D
6E 00 2C 00030
FC14 CD 00035
6E 00 2C 00038
0220 CE 0003D
6E 00 2C 00040
FD24 CD 00045
6E 00 2C 00048
0228 CE 0004D
6E 00 2C 00050
18 AE 00055
FB AD 02 90 00057
FD20 CD FC1C CD 9E 0005B
FC18 CD 0240 CE 9E 0005D
0224 CE 20 AE 9E 00064
20 AE 9E 0006B
0C AE 94 00071
10 AE 01 D0 00074
68 01 DD 00077
04 01 FB 0007B
10 AE 50 E9 0007D
68 01 D0 0007F
04 01 FB 00080
10 AE 01 D0 00083
68 01 9F 00087 1\$:
04 01 FB 0008A
10 AE 50 E9 0008D
68 02 D0 00090
04 01 9F 00094 2\$:
10 AE 01 FB 00097
68 01 9F 0009A
04 01 FB 0009D
10 AE 03 D0 000A1 3\$:
68 01 9F 000A4
04 01 FB 000A7
10 AE 04 D0 000AA 4\$:
68 01 9F 000AE
04 01 FB 000B2
10 AE 50 E9 000B5
00A0 05 D0 000B8 5\$:
C9 9F 000BC

.EXTRN LIB\$PUT OUTPUT, SHOW\$ OBJLOCKED
.EXTRN SYSS\$CHANGE_ACL, LIBSSIGNAL
.EXTRN SYSS\$OPEN, SYSS\$GETDVI
.EXTRN SYSS\$FAO, SYSS\$FORMAT_ACL
.PSECT \$CODE\$,NOWRT,2
.ENTRY SHOW_ACL, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0091
R11
MOVAB SYSS\$CHANGE_ACL, R11
MOVAB LIBSSIGNAL, R10
MOVAB P.AAP, R9
MOVAB CLISP\$PRESENT, R8
MOVAB -2092(SP), SP
MOVCS #0, (SP), #0, #8, OBJECT_NAME 0145
MOVCS #0, (SP), #0, #8, ACE_BIN_DESC 0146
MOVCS #0, (SP), #0, #8, ACE_TXT_DESC 0147
MOVCS #0, (SP), #0, #8, OUTPUT_DESC 0148
MOVCS #0, (SP), #0, #36, GETDVI_ARGS 0149
MOVCS #0, (SP), #0, #24, ATR_ARGLIST 0150
MOVCS #0, (SP), #0, #8, IO_STATUS 0151
MOVB #2, OBJECT_NAME+3 0153
CLRL OBJECT_CHAN 0154
MOVAB ACE_STORAGE, ACE_BIN_DESC+4 0155
MOVAB ACE_TEXT, ACE_TXT_DESC+4 0156
MOVAB OUTPUT_TEXT, OUTPUT_DESC+4 0157
CLRB OUTPUT_TEXT 0158
CLRL ACL_CONTEXT 0160
MOVL #1, OBJECT_TYPE 0164
PUSHL R9 0165
CALLS #1, CLISP\$PRESENT
BLBC R0, 1\$
MOVL #1, OBJECT_TYPE
PUSHAB P.AAR 0166
CALLS #1, CLISP\$PRESENT
BLBC R0, 2\$
MOVL #2, OBJECT_TYPE
PUSHAB P.AAT 0167
CALLS #1, CLISP\$PRESENT
BLBC R0, 3\$
MOVL #3, OBJECT_TYPE
PUSHAB P.AAV 0168
CALLS #1, CLISP\$PRESENT
BLBC R0, 4\$
MOVL #4, OBJECT_TYPE
PUSHAB P.AAX 0169
CALLS #1, CLISP\$PRESENT
BLBC R0, 5\$
MOVL #5, OBJECT_TYPE
PUSHAB P.AAZ 0170

		68		01	FB	000C0	CALLS	#1, CLISPRESENT		
		04		50	E9	000C3	BLBC	R0, 6\$		
	10	AE		06	D0	000C6	MOVL	#6, OBJECT_TYPE		
			00C4	C9	9F	000CA	PUSHAB	P.ABB	0171	
		68		01	FB	000CE	CALLS	#1, CLISPRESENT		
		04		50	E9	000D1	BLBC	R0, 7\$		
	10	AE		07	D0	000D4	MOVL	#7, OBJECT_TYPE		
			F8	AD	9F	000D8	PUSHAB	OBJECT_NAME	0173	
			00D4	C9	9F	000DB	PUSHAB	P.ABD		
00000000G	00			02	FB	000DF	CALLS	#2, CLISGET VALUE		
0228	CE	000A0004		8F	D0	000E6	MOVL	#655364, ATR_ARGLIST	0178	
022C	CE	04		AE	9E	000EF	MOVAB	ACL_LOCKID, ATR_ARGLIST+4	0179	
				7E	7C	000F5	CLRQ	-(SP)	0183	
				7E	D4	000F7	CLRL	-(SP)		
			0234	CE	9F	000F9	PUSHAB	ATR_ARGLIST		
			F8	AD	9F	000FD	PUSHAB	OBJECT_NAME		
			24	AE	9F	00100	PUSHAB	OBJECT_TYPE		
				57	DD	00103	PUSHL	OBJECT_CHAN		
		6B		07	FB	00105	CALLS	#7, SYSSCHANGE_ACL		
		56		50	D0	00108	MOVL	R0, STATUS		
		13		56	E8	0010B	BLBS	STATUS, 9\$	0184	
000009B8	8F			56	D1	0010E	CMPL	STATUS, #2488	0187	
				07	12	00115	BNEQ	8\$		
	56	00000000G		8F	D0	00117	MOVL	#SHOW\$_OBJLOCKED, STATUS		
			01DF	31	0011E	8\$:	BRW	23\$	0188	
	01	10		AE	D1	00121	9\$:	CMPL	OBJECT_TYPE, #1	0194
				03	13	00125	BEQL	10\$		
			00B3	31	00127		BRW	15\$		
0050	8F	00		00	2C	0012A	10\$:	MOVC5	#0, (SP), #0, #80, \$RMS_PTR	0203
			A8	AD		00131				
	A8	AD	5003	8F	B0	00133	MOVW	#20483, \$RMS_PTR		
	AC	AD	00020000	8F	D0	00139	MOVL	#131072, \$RMS_PTR+4		
	BE	AD	4302	8F	B0	00141	MOVW	#17154, \$RMS_PTR+22		
	C7	AD		02	90	00147	MOVW	#2, \$RMS_PTR+31		
	D0	AD	FF48	CD	9E	0014B	MOVAB	OBJECT_NAME, \$RMS_PTR+40		
	D4	AD	FC	AD	D0	00151	MOVL	OBJECT_NAME+4, \$RMS_PTR+44		
	DC	AD	F8	AD	90	00156	MOVW	OBJECT_NAME, \$RMS_PTR+52		
0060	8F	00		00	2C	0015B	MOVC5	#0, (SP), #0, #96, \$RMS_PTR	0208	
			FF48	CD		00162				
	FF48	CD	6002	8F	B0	00165	MOVW	#24578, \$RMS_PTR		
	FF4A	CD		01	8E	0016C	MNEGB	#1, \$RMS_PTR+2		
	FF4C	CD	FD48	CD	9E	00171	MOVAB	OBJECT_RES_NAME, \$RMS_PTR+4		
	FF52	CD		01	8E	00178	MNEGB	#1, \$RMS_PTR+10		
	FF54	CD	FE48	CD	9E	0017D	MOVAB	OBJECT_EXP_NAME, \$RMS_PTR+12		
			A8	AD	9F	00184	PUSHAB	OBJECT_FAB	0209	
00000000G	00			01	FB	00187	CALLS	#1, SYSSOPEN		
	56			50	D0	0018E	MOVL	R0, STATUS		
	50		FF4B	CD	9A	00191	MOVZBL	OBJECT_NAME+3, R0	0213	
				0C	13	00196	BEQL	11\$		
	F8	AD		50	B0	00198	MOVW	R0, OBJECT_NAME	0216	
	FC	AD	FF4C	CD	D0	0019C	MOVL	OBJECT_NAME+4, OBJECT_NAME+4	0217	
				1D	11	001A2	BRB	13\$	0213	
		50	FF53	CD	9A	001A4	11\$:	MOVZBL	OBJECT_NAME+11, R0	0219
				0C	13	001A9	BEQL	12\$		
	F8	AD		50	B0	001AB	MOVW	R0, OBJECT_NAME	0222	
	FC	AD	FF54	CD	D0	001AF	MOVL	OBJECT_NAME+12, OBJECT_NAME+4	0223	
				0A	11	001B5	BRB	13\$	0219	

F8	AD	DC	AD	9B	001B7	12\$:	MOVZBW	OBJECT_FAB+52, OBJECT_NAME	0227
FC	AD	D4	AD	D0	001BC		MOVL	OBJECT_FAB+44, OBJECT_NAME+4	0228
	15		56	E8	001C1	13\$:	BLBS	STATUS, 14\$	0233
	7E	B0	AD	7D	001C4		MOVQ	OBJECT_FAB+8, -(SP)	0237
		F8	AD	9F	001C8		PUSHAB	OBJECT_NAME	
			01	DD	001CB		PUSHL	#1	
	6A	007810B4	8F	DD	001CD		PUSHL	#7868596	
			05	FB	001D3		CALLS	#5, LIB\$SIGNAL	
			012C	31	001D6		BRW	24\$	0238
	57	B4	AD	D0	001D9	14\$:	MOVL	OBJECT_FAB+12, OBJECT_CHAN	0240
FD24	CD	00040004	8F	D0	001DD	15\$:	MOVL	#262148, GETDVI_ARGS	0246
FD28	CD	08	AE	9E	001E6		MOVAB	DEVICE_CLASS, GETDVI_ARGS+4	0247
FD30	CD	00080004	8F	D0	001EC		MOVL	#524292, GETDVI_ARGS+12	0249
FD34	CD	14	AE	9E	001F5		MOVAB	DISPLAY_WIDTH, GETDVI_ARGS+16	0250
			7E	7C	001FB		CLRQ	-(SP)	0254
			7E	D4	001FD		CLRL	-(SP)	
		24	AE	9F	001FF		PUSHAB	IO STATUS	
	FD24		CD	9F	00202		PUSHAB	GETDVI_ARGS	
	00E8		C9	9F	00206		PUSHAB	P.ABF	
			7E	7C	0020A		CLRQ	-(SP)	
00000000G	00		08	FB	0020C		CALLS	#8, SYSS\$GETDVI	
	56		50	D0	00213		MOVL	R0, STATUS	
	04		56	E9	00216		BLBC	STATUS, 16\$	0255
	56	18	AE	3C	00219		MOVZWL	IO STATUS, STATUS	
	03		56	E8	0021D	16\$:	BLBS	STATUS, 17\$	0256
			00DD	31	00220		BRW	23\$	
00000042	8F	08	AE	D1	00223	17\$:	CMPL	DEVICE_CLASS, #66	0262
			05	13	0022B		BEQL	18\$	
14	AE	84	8F	9A	0022D		MOVZBL	#132, DISPLAY_WIDTH	
0228	CE	000900FF	8F	D0	00232	18\$:	MOVL	#590079, ATR_ARGLIST	0271
022C	CE	FC1C	CD	9E	0023B		MOVAB	ACE_STORAGE, ATR_ARGLIST+4	0272
		0C	AE	9F	00242		PUSHAB	ACL_CONTEXT	0277
			7E	7C	00245		CLRQ	-(SP)	
	0234		CE	9F	00247		PUSHAB	ATR_ARGLIST	
	F8		AD	9F	0024B		PUSHAB	OBJECT_NAME	
	24		AE	9F	0024E		PUSHAB	OBJECT_TYPE	
			57	DD	00251		PUSHL	OBJECT_CHAN	
	6B		07	FB	00253		CALLS	#7, SYSS\$CHANGE_ACL	
	56		50	D0	00256		MOVL	R0, STATUS	
000009D0	8F		56	E8	00259		BLBS	STATUS, 21\$	0278
			56	D1	0025C		CMPL	STATUS, #2512	0281
			07	13	00263		BEQL	19\$	
000009E0	8F		56	D1	00265		CMPL	STATUS, #2528	
			03	12	0026C	19\$:	BNEQ	20\$	
			00AB	31	0026E		BRW	26\$	
			56	DD	00271	20\$:	PUSHL	STATUS	0282
		F8	AD	9F	00273		PUSHAB	OBJECT_NAME	
			01	DD	00276		PUSHL	#1	
		007810B4	8F	DD	00278		PUSHL	#7868596	
	6A		04	FB	0027E		CALLS	#4, LIB\$SIGNAL	
	50	107810B4	8F	D0	00281		MOVL	#276304052, R0	0283
			04		00288		RET		
A3	FC1F	CD	02	E0	00289	21\$:	BBS	#2, ACE_STORAGE+3, 18\$	0288
			20	AE	95		TSTB	OUTPUT_TEXT	0295
			35	12	00292		BNEQ	22\$	
	0220	CE	8F	B0	00294		MOVW	#512, OUTPUT_DESC	0298
		0200	7E	D4	0029B		CLRL	-(SP)	0304

		F8	AD	9F	0029D	PUSHAB	OBJECT_NAME	:	
	50	18	AE	D0	002A0	MOVL	OBJECT_TYPE, R0	:	
		D0	A940	DD	002A4	PUSHL	OBJ_TYPE_NAMES[R0]	:	
		022C	CE	9F	002A8	PUSHAB	OUTPUT_DESC	:	
		0230	CE	9F	002AC	PUSHAB	OUTPUT_DESC	:	
		0120	C9	9F	002B0	PUSHAB	P.ABH	:	
00000000G	00		06	FB	002B4	CALLS	#6, SYSS\$FA0	:	
	56		50	D0	002BB	MOVL	R0, STATUS	:	0305
		0220	CE	9F	002BE	PUSHAB	OUTPUT_DESC	:	
00000000G	00		01	FB	002C2	CALLS	#1, LIB\$PUT_OUTPUT	:	
FD1C	CD	FC1C	CD	9B	002C9	MOVZBW	ACE_STORAGE, ACE_BIN_DESC	:	0310
FC14	CD	0200	8F	B0	002D0	MOVW	#512, ACE_TXT_DESC	:	0311
			7E	D4	002D7	CLRL	-(SP)	:	0317
04	AE		0A	D0	002D9	MOVL	#10, 4(SP)	:	
		04	AE	9F	002DD	PUSHAB	4(SP)	:	
		012C	C9	9F	002E0	PUSHAB	P.ABJ	:	
		20	AE	9F	002E4	PUSHAB	DISPLAY_WIDTH	:	
		FC14	CD	9F	002E7	PUSHAB	ACE_TXT_DESC	:	
		FC14	CD	9F	002EB	PUSHAB	ACE_TXT_DESC	:	
		FD1C	CD	9F	002EF	PUSHAB	ACE_BIN_DESC	:	
00000000G	00		07	FB	002F3	CALLS	#7, SYSS\$FORMAT_ACL	:	
	56		50	D0	002FA	MOVL	R0, STATUS	:	0318
	0E		56	E8	002FD	BLBS	STATUS, 25\$	:	0321
			56	DD	00300	PUSHL	STATUS	:	
	6A		01	FB	00302	CALLS	#1, LIB\$SIGNAL	:	
50	56	10000000	8F	C9	00305	BISL3	#268435456, STATUS, R0	:	0322
			04	0030D	RET			:	
		FC14	CD	9F	0030E	PUSHAB	ACE_TXT_DESC	:	0324
00000000G	00		01	FB	00312	CALLS	#1, LIB\$PUT_OUTPUT	:	
			FF16	31	00319	BRW	18\$	:	0267
000009D0	8F		56	D1	0031C	CMPL	STATUS, #2512	:	0330
			06	12	00323	BNEQ	27\$	:	
	50	09D0	8F	3C	00325	MOVZWL	#2512, R0	:	0332
			04	0032A	RET			:	
	50		01	D0	0032B	MOVL	#1, R0	:	
			04	0032E	RET			:	0334

; Routine Size: 815 bytes, Routine Base: \$CODE\$ + 0000

:	337	0335	1
:	338	0336	1 END
:	339	0337	0 ELUDOM

PSECT SUMMARY

:	Name	Bytes	Attributes
:	\$PLITS	504	NOVEC,NOWRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)
:	\$CODE\$	815	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)

SHOW\$ACL
V04-000

1 4
16-Sep-1984 00:07:24
14-Sep-1984 11:52:36

VAX-11 Bliss-32 V4.0-742
[ACLEDT.SRC]SHOWACL.B32;1

Page 14
(3)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	103	0	1000	00:01.9

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

COMMAND QUALIFIERS

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

; Size: 815 code + 504 data bytes
; Run Time: 00:17.4
; Elapsed Time: 00:52.9
; Lines/CPU Min: 1164
; Lexemes/CPU-Min: 26823
; Memory Used: 257 pages
; Compilation Complete

0005 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY