

MOU
V04[illegible]

A 10x10 grid of characters forming the letters 'L', 'I', and 'S'. The grid is composed of the following characters:

- L**: Located in the first column, rows 1-10.
- I**: Located in the fifth column, rows 1-10.
- S**: Located in the ninth column, rows 1-10.

The characters are arranged in a pattern that resembles a stylized 'LIS' or 'LIS'.


```
0001 0 MODULE MOUPAR (  
0002 0     LANGUAGE (BLISS32),  
0003 0     IDENT = 'V04-0U0'  
0004 0 ) =  
0005 1 BEGIN  
0006 1  
0007 1  
0008 1 *****  
0009 1 *  
0010 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
0011 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
0012 1 *  ALL RIGHTS RESERVED.  
0013 1 *  
0014 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
0015 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
0016 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
0017 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
0018 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
0019 1 *  TRANSFERRED.  
0020 1 *  
0021 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
0022 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
0023 1 *  CORPORATION.  
0024 1 *  
0025 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
0026 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
0027 1 *  
0028 1 *  
0029 1 *****  
0030 1  
0031 1 ++  
0032 1  
0033 1 FACILITY: MOUNT Utility Structure Level 1  
0034 1  
0035 1 ABSTRACT:  
0036 1  
0037 1     This module contains the data base and utilities used to acquire the  
0038 1     MOUNT command line from the CLI parser.  
0039 1  
0040 1 ENVIRONMENT:  
0041 1  
0042 1     STARLET operating system, including privileged system services  
0043 1     and internal exec routines.  
0044 1  
0045 1 --  
0046 1  
0047 1  
0048 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 29-Sep-1977 16:58  
0049 1  
0050 1 MODIFIED BY:  
0051 1  
0052 1     V03-013 HH0041 Hai Huang 24-Jul-1984  
0053 1     Remove REQUIRE 'LIBD$:[VMSLIB.OBJ]MOUNTMSG.B32'.  
0054 1  
0055 1     V03-012 DAS0001 David Solomon 10-Jul-1984  
0056 1     Add support for MNTSV_NOREBUILD (/NOREBUILD qualifier).  
0057 1
```

58	0058	1	V03-011	HH0009	Hai Huang	27-Mar-1984
59	0059	1		Add security auditing support, i.e. move MOUNT_FLAGS to		
60	0060	1		\$GLOBALS psect.		
61	0061	1	V03-010	HH0004	Hai Huang	09-Mar-1984
62	0062	1		Add cluster-wide mount support (/CLUSTER qualifier).		
63	0063	1	V03-009	DAS0001	David Solomon	29-Nov-1983
64	0064	1		Add support for specifying maximum journal record size		
65	0065	1		with a new keyword, /JOURNAL=(RECORD_SIZE=n).		
66	0066	1	V03-008	MMD0189	Meg Dumont,	7-Jul-1983 10:00
67	0067	1		Make the default for AVL/AVR the same from the DCL call		
68	0068	1		and from the system service call.		
69	0069	1	V03-007	MMD0114	Meg Dumont,	29-Mar-1983 0:38
70	0070	1		Add support for new qualifiers for AVL, AVR and VMS protection		
71	0071	1	V03-006	STJ48132	Steven T. Jeffreys,	09-Oct-1982
72	0072	1		Initialize MOUNT_FLAGS before using it.		
73	0073	1	V03-005	STJ0317	Steven T. Jeffreys,	15-Aug-1982
74	0074	1		Added support for journalling qualifers.		
75	0075	1	V03-004	STJ0304	Steven T. Jeffreys,	18-May-1982
76	0076	1		Added support for /NOUNLOAD qualifier.		
77	0077	1	V03-003	STJ0260	Steven T. Jeffreys,	14-Apr-1982
78	0078	1		Removed the device list inspection code added in update V02-016		
79	0079	1	V03-002	STJ0250	Steven T. Jeffreys,	01-Apr-1982
80	0080	1		- Check for invalid device name status from \$GETDEV.		
81	0081	1	V03-001	STJ0238	Steven T. Jeffreys,	17-Mar-1982
82	0082	1		- Check for CMKRNL privilege if /PROCESSOR=<filename> specified.		
83	0083	1	V02-018	STJ0230	Steven T. Jeffreys,	01-Mar-1982
84	0084	1		- If user is trying to access a non-existent drive,		
85	0085	1		return a status to indicate such. This is an improvement		
86	0086	1		to update #16 to this module.		
87	0087	1		- Close buffer probe protection hole.		
88	0088	1	V02-017	STJ0225	Steven T. Jeffreys,	17-Feb-1982
89	0089	1		Fix incorrect probe of the users input parameters.		
90	0090	1	V02-016	STJ0217	Steven t. Jeffreys,	15-Feb-1982
91	0091	1		Sweep through the device list to make sure that each		
92	0092	1		specified device exists and is file-oriented.		
93	0093	1	V02-015	STJ0200	Steven T. Jeffreys,	11-Feb-1982
94	0094	1		Add support for the /COMMENT qualifier.		
95	0095	1	V02-014	STJ0185	Steven T. Jeffreys,	25-Jan-1982
96	0096	1		- Removed privilege restrictions for /MOUNT_VERIFICATION.		
97	0097	1		- Zero GLOBAL and OWN storage to guaranty restartability.		
98	0098	1	V02-013	STJ0173	Steven T. Jeffreys,	13-Jan-1982
99	0099	1				
100	0100	1				
101	0101	1				
102	0102	1				
103	0103	1				
104	0104	1				
105	0105	1				
106	0106	1				
107	0107	1				
108	0108	1				
109	0109	1				
110	0110	1				
111	0111	1				
112	0112	1				
113	0113	1				
114	0114	1				


```

: 115      0115 1  Modified to use the new $MOUNT interface.
: 116      0116 1
: 117      0117 1
: 118      0118 1 V02-012 STJ0157 Steven T. Jeffreys, 04-Jan-1982
: 119      0119 1 Added support for the /OVERRIDE=LOCK, /NOCACHE, and /MOUNTVER
: 120      0120 1 qualifiers. Changed reference of OPT_SIGNAL to OPT_MESSAGE.
: 121      0121 1
: 122      0122 1 V02-011 STJ0152 Steven T. Jeffreys, 02-Jan-1981
: 123      0123 1 Extensive rewrite to support the $MOUNT system service.
: 124      0124 1
: 125      0125 1 V02-010 STJ0148 Steven T. Jeffreys, 16-Dec-1981
: 126      0126 1 Add a "." to the front of the VOLSET parameter reference.
: 127      0127 1
: 128      0128 1 V02-009 BLS0080 Benn Schreiber, 14-Sep-1981
: 129      0129 1 Reference 'BIND' rather than 'VOLSET' for /BIND qualifier
: 130      0130 1
: 131      0131 1 V02-008 STJ0109 Steven T. Jeffreys, 28-Aug-1981
: 132      0132 1 Liberal rewrite as part of $MOUNT support.
: 133      0133 1
: 134      0134 1 V02-007 STJ0036 Steven T. Jeffreys, 11-May-1981
: 135      0135 1 Added support for /ASSIST qualifier.
: 136      0136 1
: 137      0137 1 V02-006 ACG0167 Andrew C. Goldstein, 18-Apr-1980 13:38
: 138      0138 1 Previous revision history moved to MOUNT.REV
: 139      0139 1
: 140      0140 1
: 141      0141 1 LIBRARY 'SYSS$LIBRARY:LIB.L32';
: 142      0142 1 REQUIRE 'SRC$:MOUDEF.B32';
: 143      0674 1 LIBRARY 'SYSS$LIBRARY:CLIMAC.L32';
: 144      0675 1 LIBRARY 'SYSS$LIBRARY:TPAMAC.L32';

```

```
146 0676 1 !+
147 0677 1
148 0678 1
149 0679 1 Impure data area. This area contains the MOUNT parameters extracted from
150 0680 1 the command line by the associated parsing routines.
151 0681 1 !-
152 0682 1
153 0683 1
154 0684 1 GLOBAL
155 0685 1
156 0686 1 GLOBAL_START : VECTOR [0] ! mark start of global storage
157 0687 1 MOUNT_OPTIONS : BITVECTOR [64] ! option flags
158 0688 1 MOUNT_FLAGS : BBLOCK [4], ! Mount option flags
159 0689 1 PROTECTION, ! value of /PROTECTION switch
160 0690 1 OWNER_UIC, ! value of /OWNER UIC switch
161 0691 1 USER_UIC, ! value of /USER UIC switch
162 0692 1 EXTENSION, ! value of /EXTENSION switch
163 0693 1 WINDOW, ! value of /WINDOW switch
164 0694 1 ACCESSED, ! value of /ACCESSED switch
165 0695 1 BLOCKSIZE, ! value of /BLOCK switch
166 0696 1 RECORDSZ, ! value of /RECORD switch
167 0697 1 FID_CACHE, ! space to allocate for file ID cache
168 0698 1 EXT_CACHE, ! space to allocate for extent cache
169 0699 1 QUO_CACHE, ! space to allocate for quota cache
170 0700 1 EXT_LIMIT, ! limit of disk free space to cache
171 0701 1 DEVICE_COUNT, ! number of devices specified
172 0702 1 LABEL_COUNT, ! number of volume labels specified
173 0703 1 VID_STRING : VECTOR [2], ! descriptor of VISUAL ID string
174 0704 1 COMMENT_STRING : VECTOR [2], ! descriptor of COMMENT string
175 0705 1 ACP_STRING : VECTOR [2], ! descriptor of ACP device or name string
176 0706 1 LOG_NAME : VECTOR [2], ! descriptor of logical name string
177 0707 1 STRUCT_NAME : VECTOR [2], ! descriptor of volume set name
178 0708 1 ! (value of /BIND qualifier)
179 0709 1 DRIVE_COUNT : VECTOR [DEVMAX], ! value of /DRIVES switch
180 0710 1 DEVICE_STRING : VECTOR [DEVMAX*2], ! descriptors of device name strings
181 0711 1 LABEL_STRING : VECTOR [LABMAX*2], ! descriptors of volume label strings
182 0712 1 JRNL_SIZE, ! Recovery Unit Journal (RUJ) initial size
183 0713 1 JRNL_EXTEND, ! RUJ default file extension size
184 0714 1 JRNL_QUOTA, ! RUJ byte quota per recovery unit
185 0715 1 JRNL_RECORD_SIZE, ! RUJ maximum record size
186 0716 1 GLOBAL_END : VECTOR [0]; ! Mark end of global storage
187 0717 1 LITERAL
188 0718 1 COMMENT_SIZE = 80, ! maximum length of a user comment
189 0719 1 BUFFER_SIZE = 63, ! maximum length of a string buffer
190 0720 1 ITEM_SIZE = 12; ! length of each item descriptor
191 0721 1
192 0722 1 OWN
193 0723 1 OWN_START : VECTOR [0], ! Mark start of OWN storage
194 0724 1 DEVICE_BUFFER : BBLOCK [BUFFER_SIZE*DEVMAX], ! Buffer for all device names
195 0725 1 LABEL_BUFFER : BBLOCK [BUFFER_SIZE*DEVMAX], ! Buffer for all volume labels
196 0726 1 LOG_NAME_BUFFER : BBLOCK [BUFFER_SIZE], ! Buffer for the volume logical name
197 0727 1 ACP_NAME_BUFFER : BBLOCK [BUFFER_SIZE], ! Buffer for the user-specified ACP name
198 0728 1 VOLSET_BUFFER : BBLOCK [BUFFER_SIZE], ! Buffer for the volume set name
199 0729 1 COMMENT_BUFFER : BBLOCK [COMMENT_SIZE], ! Buffer for the user comment
200 0730 1 TPARSE_BLOCK : BBLOCK [TPASK_LENGTH0], ! TPARSE control block
201 0731 1 OWN_END : VECTOR [0]; ! Mark end of OWN storage
202 0732 1
```



```

: 203      0733 1 LITERAL
: 204      0734 1
: 205      0735 1      | Define the lengths of the GLOBAL and OWN storage areas.
: 206      0736 1
: 207      0737 1      GLOBAL_LENGTH = GLOBAL_END - GLOBAL_START,
: 208      0738 1      OWN_LENGTH   = OWN_END - OWN_START;
: 209      0739 1
: 210      0740 1 FORWARD ROUTINE
: 211      0741 1      COPY_ITEM;                                ! Copy an item to internal storage
: 212      0742 1
: 213      0743 1 EXTERNAL
: 214      0744 1      DEVCHAR_DESC : VECTOR,
: 215      0745 1      DEVICE_CHAR  : BBLOCK,
: 216      0746 1      NEWLINE      : BBLOCK,
: 217      0747 1      CTLSGL_PHD   : ADDRESSING_MODE (GENERAL);
: 218      0748 1
: 219      0749 1 FORWARD
: 220      0750 1      ACP_KTB       : VECTOR [0],                ! TPARSE table label.
: 221      0751 1      ACP_STB      : VECTOR [0];                ! TPARSE table label.
: 222      0752 1
: 223      0753 1 MACRO
: 224      0754 1      |
: 225      0755 1      | Symbolic offsets into an item descriptor
: 226      0756 1      |
: 227      0757 1      LENGTH       = 0, 0, 16, 0%,              ! length of item (in bytes)
: 228      0758 1      CODE        = 2, 0, 16, 0%,              ! item code
: 229      0759 1      ADDRESS     = 4, 0, 32, 0%,              ! item address
: 230      0760 1      UNUSED      = 8, 0, 32, 0%;              ! reserved for future use
: 231      0761 1
```

```
233 0762 1 GLOBAL ROUTINE CHECK_PARAMS (ITEM_LIST) =
234 0763 1 ++
235 0764 1 Functional description:
236 0765 1
237 0766 1 This routine is responsible for validating the parameters passed
238 0767 1 by the $MOUNT system service call, and copyin them to internal
239 0768 1 storage areas. The validation consits of:
240 0769 1
241 0770 1 - Probing each parameter for read access.
242 0771 1 - Copying each parameter to an internal area.
243 0772 1 - Sanity checks on the parameter values.
244 0773 1
245 0774 1 Input:
246 0775 1
247 0776 1 None.
248 0777 1
249 0778 1 Implicit Input:
250 0779 1
251 0780 1 None.
252 0781 1
253 0782 1 Output:
254 0783 1
255 0784 1 None.
256 0785 1
257 0786 1 Implicit output:
258 0787 1
259 0788 1 Mount's internal data storage is initialized.
260 0789 1
261 0790 1 Side effects:
262 0791 1
263 0792 1 None.
264 0793 1
265 0794 1 Routine value:
266 0795 1
267 0796 1 SSS_NORMAL : Normal successful completion
268 0797 1 SSS_ACCVIO : A parameter was not readable
269 0798 1 SSS_BADPARAM : Bad parameter value
270 0799 1 MOUN$_CONFQUAL : The command has conflicting qualifiers
271 0800 1
272 0801 1 --
273 0802 1
274 0803 2 BEGIN
275 0804 2
276 0805 2 BUILTIN
277 0806 2 AP, : Arguement pointer
278 0807 2 PROBER; : Probe for read access
279 0808 2
280 0809 2 EXTERNAL ROUTINE
281 0810 2 LIB$TPARSE : ADDRESSING_MODE (GENERAL); ! Finite state parser
282 0811 2
283 0812 2 LOCAL
284 0813 2 PHD : REF BBLOCK, : Process PHD
285 0814 2 ITEM : REF BBLOCK, : Pointer to item list
286 0815 2 PTR : LONG, : Temporary storage
287 0816 2 STR_DESCRIPTOR : REF BBLOCK, : Temporary descriptor
288 0817 2 JRNC_ITEM_SEEN : LONG, : A boolean
289 0818 2 DENSITY : LONG, : Tape volume density
```



```
290      STATUS      : LONG;      ! Store routine status code
291
292      Zero the GLOBAL and OWN storage areas to guaranty the
293      restartability of the code.
294
295      CH$FILL (0, GLOBAL_LENGTH, GLOBAL_START);
296      CH$FILL (0, OWN_LENGTH, OWN_START);
297      MOUNT_FLAGS [0, 0, 32, 0] = 0;
298
299      Initialize the TPARSE control block.
300
301      TPARSE_BLOCK [TPASL_COUNT]      = TPASK_COUNT0;
302      TPARSE_BLOCK [TPASL_OPTIONS]    = (TPASM_BLANKS OR TPASM_ABBREV);
303
304      *****
305      Interpret each item in the item list. After doing some
306      sanity checks for each parameter, copy it to the internal
307      storage area. This is done to protect the data from corruption
308      by malicious users. The item list is terminated by an item
309      descriptor with an item code of 0. Items with a zero length are
310      rejected.
311
312      Since a given item may appear in the list more than once, we must be
313      careful to reset any item context for each version of the item. A
314      good example of where this is necessary is for the MNTS_DENSITY item.
315      *****
316
317      MOUNT_OPTIONS = MOUNT_OPTIONS+4 = 0;
318      STATUS = 1;
319      JRNL_ITEM_SEEN = 0;
320      DEVICE_COUNT = 0;
321      LABEL_COUNT = 0;
322      ITEM = .ITEM_LIST;
323      IF NOT PROBER (%REF (0), %REF (4), ITEM [0, 0, 32, 0]) ! The first 4 bytes must be readable
324      THEN
325          RETURN (SS$_ACCVIO);
326
327      WHILE .STATUS AND (.ITEM [CODE] NEQ 0) DO
328      BEGIN
329          PROBE the next 12 bytes, which includes the remainder of this item
330          descriptor and the first longword of the next one. If duplicate
331          items are declared, then the value last encountered will be used.
332          The exception to this are device names and volume labels.
333
334          As new items are defined, a corresponding entry must be made in
335          the select table defined below.
336
337          IF NOT PROBER (%REF (0), %REF (ITEM_SIZE), .ITEM)
338          THEN
339              STATUS = SS$_ACCVIO
340          ELSE
341              BEGIN
```

```

: 347 0876 5 STATUS = (SELECTONE .ITEM [CODE] OF
: 348 0877 5 SET
: 349 0878 5
: 350 0879 5 The following items are character strings.
: 351 0880 5
: 352 0881 6 [MNT$_DEVNAM] : (IF .DEVICE_COUNT LSS DEVMAX
: 353 0882 6 THEN
: 354 0883 7 BEGIN
: 355 0884 7 MOUNT_OPTIONS [OPT_DEVICE] = 1;
: 356 0885 7 DEVICE_COUNT = .DEVICE_COUNT + 1;
: 357 0886 7 DEVICE_STRING [( .DEVICE_COUNT-1)*2] = MIN (.ITEM [LENGTH], BUF
: 358 0887 7 DEVICE_STRING [( .DEVICE_COUNT-1)*2)+1] = DEVICE_BUFFER+(BUFFE
: 359 0888 7 COPY_ITEM (.ITEM, BUFFER_SIZE, .DEVICE_STRING [( .DEVICE_COUNT
: 360 0889 7 END
: 361 0890 6 ELSE
: 362 0891 5 MOUN$_MAXDEV);
: 363 0892 5
: 364 0893 6 [MNT$_VOLNAM] : (IF .LABEL_COUNT LSS DEVMAX
: 365 0894 6 THEN
: 366 0895 7 BEGIN
: 367 0896 7 MOUNT_OPTIONS [OPT_LABEL] = 1;
: 368 0897 7 LABEL_COUNT = .LABEL_COUNT + 1;
: 369 0898 7 LABEL_STRING [( .LABEL_COUNT-1)*2] = MIN (.ITEM [LENGTH], BUFFE
: 370 0899 7 LABEL_STRING [( .LABEL_COUNT-1)*2)+1] = LABEL_BUFFER + (BUFFER
: 371 0900 7 COPY_ITEM (.ITEM, BUFFER_SIZE, .LABEL_STRING [( .LABEL_COUNT-1
: 372 0901 7 END
: 373 0902 6 ELSE
: 374 0903 5 MOUN$_MAXLAB);
: 375 0904 5
: 376 0905 6 [MNT$_LOGNAM] : BEGIN
: 377 0906 6 MOUNT_OPTIONS [OPT_LOG_NAME] = 1; ! Mark the logical n
: 378 0907 6 LOG_NAME [0] = MIN (.ITEM [LENGTH], BUFFER_SIZE);
: 379 0908 6 LOG_NAME [1] = LOG_NAME_BUFFER;
: 380 0909 6 STATUS = COPY_ITEM (.ITEM, BUFFER_SIZE, LOG_NAME_BUFFER);
: 381 0910 6
: 382 0911 6 Scan for a trailing or imbedded colon. If found,
: 383 0912 6 use the string preceding the colon.
: 384 0913 6
: 385 0914 6 PTR = CH$FIND CH (.LOG_NAME [0], .LOG_NAME [1], ':');
: 386 0915 6 IF NOT CH$FAIL (.PTR)
: 387 0916 6 THEN
: 388 0917 6 LOG_NAME [0] = .PTR - .LOG_NAME [1];
: 389 0918 6 .STATUS
: 390 0919 5 END;
: 391 0920 5
: 392 0921 6 [MNT$_PROCESSOR] : BEGIN
: 393 0922 6 ACP_STRING [0] = MIN (.ITEM [LENGTH], BUFFER_SIZE);
: 394 0923 6 ACP_STRING [1] = ACP_NAME_BUFFER;
: 395 0924 6 COPY_ITEM (.ITEM, BUFFER_SIZE, ACP_NAME_BUFFER)
: 396 0925 5 END;
: 397 0926 5
: 398 0927 6 [MNT$_VOLSET] : BEGIN
: 399 0928 6 STRUCT_NAME [0] = MIN (.ITEM [LENGTH], BUFFER_SIZE);
: 400 0929 6 STRUCT_NAME [1] = VOLSET_BUFFER;
: 401 0930 6 MOUNT_OPTIONS [OPT_BIND] = 1;
: 402 0931 6 COPY_ITEM (.ITEM, BUFFER_SIZE, VOLSET_BUFFER)
: 403 0932 5 END;
```


461	0990	6	ELSE
462	0991	6	MOUNT_OPTIONS [OPT_CACHE] = 1;
463	0992	6	.STATUS
464	0993	5	END;
465	0994	5	
466	0995	5	[MNT\$_LIMIT] : COPY_ITEM (.ITEM, 4, EXT_LIMIT);
467	0996	5	
468	0997	6	[MNT\$_OWNER] : BEGIN
469	0998	6	MOUNT_OPTIONS [OPT_OWNER_UIC] = 1;
470	0999	6	COPY_ITEM (.ITEM, 4, OWNER_UIC)
471	1000	5	END;
472	1001	5	
473	1002	6	[MNT\$_VPROT] : BEGIN
474	1003	6	MOUNT_OPTIONS [OPT_PROTECTION] = 1;
475	1004	6	COPY_ITEM (.ITEM, 4, PROTECTION)
476	1005	5	END;
477	1006	5	
478	1007	6	[MNT\$_QUOTA] : BEGIN
479	1008	6	STATUS = COPY_ITEM (.ITEM, 4, QUO_CACHE);
480	1009	6	MOUNT_OPTIONS [OPT_NOQUO_C] = 0;
481	1010	6	MOUNT_OPTIONS [OPT_CACHE] = 0;
482	1011	6	IF .QUO_CACHE LEQ 0
483	1012	6	THEN
484	1013	6	MOUNT_OPTIONS [OPT_NOQUO_C] = 1
485	1014	6	ELSE
486	1015	6	MOUNT_OPTIONS [OPT_CACHE] = 1;
487	1016	6	.STATUS
488	1017	5	END;
489	1018	5	
490	1019	6	[MNT\$_RECORDSZ] : BEGIN
491	1020	6	MOUNT_OPTIONS [OPT_RECORDSZ] = 1;
492	1021	7	IF (STATUS = COPY_ITEM (.ITEM, 4, RECORDSZ))
493	1022	6	THEN
494	1023	6	IF .RECORDSZ GTR 65534
495	1024	6	THEN
496	1025	6	STATUS = MOUN\$_SZTOOBIG;
497	1026	6	.STATUS
498	1027	5	END;
499	1028	5	
500	1029	6	[MNT\$_WINDOW] : BEGIN
501	1030	6	MOUNT_OPTIONS [OPT_WINDOW] = 1;
502	1031	6	COPY_ITEM (.ITEM, 4, WINDOW)
503	1032	5	END;
504	1033	5	
505	1034	6	[MNT\$_EXTENSION] : BEGIN
506	1035	6	MOUNT_OPTIONS [OPT_EXTENSION] = 1;
507	1036	6	COPY_ITEM (.ITEM, 4, EXTENSION)
508	1037	5	END;
509	1038	5	
510	1039	6	[MNT\$_COMMENT] : BEGIN
511	1040	6	Append a newline (<cr><lf>) to the front of the comment string.
512	1041	6	MOUNT_OPTIONS [OPT_COMMENT] = 1;
513	1042	6	COMMENT_STRING [0] = MIN (.ITEM [LENGTH], COMMENT_SIZE) + .NEWLINE
514	1043	6	COMMENT_STRING [1] = COMMENT_BUFFER;
515	1044	6	CH\$MOVE (.NEWLINE [DSC\$_LENGTH], .NEWLINE [DSC\$_POINTER], COMMEN
516	1045	6	
517	1046	6	


```

: 518      1047 6
: 519      1048 6
: 520      1049 6
: 521      1050 6
: 522      1051 5
: 523      1052 5
: 524      1053 6
: 525      1054 6
: 526      1055 6
: 527      1056 5
: 528      1057 5
: 529      1058 6
: 530      1059 6
: 531      1060 6
: 532      1061 5
: 533      1062 5
: 534      1063 6
: 535      1064 6
: 536      1065 6
: 537      1066 5
: 538      1067 5
: 539      1068 6
: 540      1069 6
: 541      1070 6
: 542      1071 5
: 543      1072 5
: 544      1073 5
: 545      1074 5
: 546      1075 4
: 547      1076 4
: 548      1077 5
: 549      1078 5
: 550      1079 5
: 551      1080 2
: 552      1081 2
: 553      1082 2
: 554      1083 2
: 555      1084 2
: 556      1085 2
: 557      1086 2

COPY_ITEM (.ITEM,
           COMMENT_SIZE-.NEWLINE [DSC$W_LENGTH],
           COMMENT_BUFFER+.NEWLINE [DSC$W_LENGTH]
           )
END;

[MNT$_JRNLSIZE] : BEGIN
                JRNL_ITEM_SEEN = 1;
                COPY_ITEM(.ITEM, 4, JRNL_SIZE)
                END;

[MNT$_JRNLEXTEND] : BEGIN
                  JRNL_ITEM_SEEN = 1;
                  COPY_ITEM(.ITEM, 4, JRNL_EXTEND)
                  END;

[MNT$_JRNLQUOTA] : BEGIN
                  JRNL_ITEM_SEEN = 1;
                  COPY_ITEM(.ITEM, 4, JRNL_QUOTA)
                  END;

[MNT$_JRNLRECORD_SIZE]: BEGIN
                       JRNL_ITEM_SEEN = 1;
                       COPY_ITEM(.ITEM, 4, JRNL_RECORD_SIZE)
                       END;

[OTHERWISE] : SS$_BADPARAM;
TES

        );
        ITEM = .ITEM + ITEM_SIZE;      ! Point to the next item in the list.
    END;
END;

! If something went awry, stop right now.
IF NOT .STATUS
THEN
    RETURN (.STATUS);
!
```

```

559 1087 *****
560 1088
561 1089 Now perform some additional sanity checks on each of the parameters.
562 1090 The ordering of the processing is not important. Note that some
563 1091 privilege checking is done here, and explicit 'no xxxx privilege'
564 1092 status codes are returned to make $MOUNT more friendly.
565 1093 *****
566 1094
567 1095
568 1096
569 1097 Set the appropriate mount option flags.
570 1098
571 1099 MOUNT_OPTIONS [OPT_CLUSTER] = .MOUNT_FLAGS [MNTSV_CLUSTER];
572 1100 MOUNT_OPTIONS [OPT_SYSTEM] = .MOUNT_FLAGS [MNTSV_SYSTEM];
573 1101 MOUNT_OPTIONS [OPT_GROUP] = .MOUNT_FLAGS [MNTSV_GROUP];
574 1102 MOUNT_OPTIONS [OPT_MOUNTVER] = NOT .MOUNT_FLAGS [MNTSV_NOMNTVER];
575 1103 MOUNT_OPTIONS [OPT_NOQUOTA] = .MOUNT_FLAGS [MNTSV_NODISKQ];
576 1104 MOUNT_OPTIONS [OPT_NOHDR3] = .MOUNT_FLAGS [MNTSV_NOHDR3];
577 1105 MOUNT_OPTIONS [OPT_SHARE] = .MOUNT_FLAGS [MNTSV_SHARE];
578 1106 MOUNT_OPTIONS [OPT_WRITE] = NOT .MOUNT_FLAGS [MNTSV_NOWRITE];
579 1107 MOUNT_OPTIONS [OPT_NOCACHE] = .MOUNT_FLAGS [MNTSV_NOCACHE];
580 1108 MOUNT_OPTIONS [OPT_OVR_LOCK] = .MOUNT_FLAGS [MNTSV_OVR_LOCK];
581 1109 MOUNT_OPTIONS [OPT_MESSAGE] = .MOUNT_FLAGS [MNTSV_MESSAGE];
582 1110 MOUNT_OPTIONS [OPT_READCHECK] = .MOUNT_FLAGS [MNTSV_READCHECK];
583 1111 MOUNT_OPTIONS [OPT_WRITECHECK] = .MOUNT_FLAGS [MNTSV_WRITECHECK];
584 1112 MOUNT_OPTIONS [OPT_WTHRU] = .MOUNT_FLAGS [MNTSV_WRITETHRU];
585 1113 MOUNT_OPTIONS [OPT_ASSIST] = NOT .MOUNT_FLAGS [MNTSV_NOASSIST];
586 1114 MOUNT_OPTIONS [OPT_FOREIGN] = .MOUNT_FLAGS [MNTSV_FOREIGN];
587 1115 MOUNT_OPTIONS [OPT_OVR_EXP] = .MOUNT_FLAGS [MNTSV_OVR_EXP];
588 1116 MOUNT_OPTIONS [OPT_OVR_ID] = .MOUNT_FLAGS [MNTSV_OVR_IDENT];
589 1117 MOUNT_OPTIONS [OPT_OVR_SETID] = .MOUNT_FLAGS [MNTSV_OVR_SETID];
590 1118 MOUNT_OPTIONS [OPT_OVR_ACC] = .MOUNT_FLAGS [MNTSV_OVR_ACCESS];
591 1119 MOUNT_OPTIONS [OPT_BLOCK] = 1;
592 1120 MOUNT_OPTIONS [OPT_NOUNLOAD] = .MOUNT_FLAGS [MNTSV_NOUNLOAD];
593 1121 MOUNT_OPTIONS [OPT_NOJRN] = .MOUNT_FLAGS [MNTSV_NOJRN];
594 1122 MOUNT_OPTIONS [OPT_NEWJRN] = .MOUNT_FLAGS [MNTSV_NEWJRN];
595 1123 MOUNT_OPTIONS [OPT_NOAUTO] = .MOUNT_FLAGS [MNTSV_NOAUTO];
596 1124 MOUNT_OPTIONS [OPT_INIT_ALL] = .MOUNT_FLAGS [MNTSV_INIT_ALL];
597 1125 MOUNT_OPTIONS [OPT_INIT_CONT] = .MOUNT_FLAGS [MNTSV_INIT_CONT];
598 1126 MOUNT_OPTIONS [OPT_OVR_VOLO] = .MOUNT_FLAGS [MNTSV_OVR_VOLO];
599 1127 MOUNT_OPTIONS [OPT_INTERCHG] = .MOUNT_FLAGS [MNTSV_INTERCHG];
600 1128 MOUNT_OPTIONS [OPT_NOREBUILD] = .MOUNT_FLAGS [MNTSV_NOREBUILD];
601 1129
602 1130
603 1131 ++
604 1132 Process the journalling qualifers. The valued parameters may only
605 1133 be specified if MNTSV_NEWJRN is specified. It is the responsibility
606 1134 of the actual journalling code to check the user supplied values for
607 1135 validity, and to apply default values where appropriate.
608 1136
609 1137 Note that MNTSV_NEWJRN and MNTSV_NOJRN are mutually exclusive.
610 1138
611 1139 IF (.MOUNT_OPTIONS [OPT_NEWJRN] AND .MOUNT_OPTIONS [OPT_NOJRN])
612 1140 OR (.JRNL_ITEM_SEEN AND NOT .MOUNT_OPTIONS [OPT_NEWJRN])
613 1141 THEN
614 1142 RETURN (MOUN$_CONFQUAL);
615 1143
```



```

616      1144      2  !++
617      1145      2  ! Parse the ACP name, if one was provided.
618      1146      2  !--
619      1147      2  !
620      1148      2  IF .ACP_STRING NEQ 0
621      1149      2  THEN
622      1150      2  BEGIN
623      1151      2  !
624      1152      2  ! Call TPARSE to interpret this parameter. All the work
625      1153      2  ! is done by the action routine.
626      1154      2  !
627      1155      2  STR_DESCRIPTOR = ACP_STRING;
628      1156      2  TPARSE_BLOCK [TPASL_STRINGCNT] = .STR_DESCRIPTOR [DSC$W_LENGTH];
629      1157      2  TPARSE_BLOCK [TPASL_STRINGPTR] = .STR_DESCRIPTOR [DSC$A_POINTER];
630      1158      2  IF NOT (STATUS = LIB$TPARSE (TPARSE_BLOCK, ACP_STB, ACP_KTB))
631      1159      2  THEN
632      1160      2  RETURN (.STATUS);
633      1161      2  !
634      1162      2  ! If the ACP string is a device or file name, copy it to internal storage.
635      1163      2  !
636      1164      2  IF .MOUNT_OPTIONS [OPT_SAMEACP]
637      1165      2  OR .MOUNT_OPTIONS [OPT_FILEACP]
638      1166      2  THEN
639      1167      2  BEGIN
640      1168      2  ACP_STRING [0] = .STR_DESCRIPTOR [DSC$W_LENGTH];
641      1169      2  ACP_STRING [1] = .STR_DESCRIPTOR [DSC$A_POINTER];
642      1170      2  END;
643      1171      2  !
644      1172      2  END;
645      1173      2  !
646      1174      2  !
647      1175      2  ! If the volume is being mounted CLUSTER without the /SYSTEM or /GROUP
648      1176      2  ! qualifier, force it to be mounted SYSTEM.
649      1177      2  !
650      1178      2  IF .MOUNT_OPTIONS [OPT_CLUSTER]
651      1179      2  AND ( NOT .MOUNT_OPTIONS [OPT_GROUP] )
652      1180      2  THEN
653      1181      2  MOUNT_OPTIONS [OPT_SYSTEM] = 1;
654      1182      2  !
655      1183      2  !
656      1184      2  ! If the volume is not being mounted SHARE, GROUP, or SYSTEM,
657      1185      2  ! then assume that the volume is being mounted NOSHARE (the default).
658      1186      2  !
659      1187      2  IF (NOT .MOUNT_OPTIONS [OPT_SHARE])
660      1188      2  AND (NOT .MOUNT_OPTIONS [OPT_GROUP])
661      1189      2  AND (NOT .MOUNT_OPTIONS [OPT_SYSTEM])
662      1190      2  THEN
663      1191      2  MOUNT_OPTIONS [OPT_NOSHARE] = 1;
664      1192      2  !
665      1193      2  !
666      1194      2  ! If the volume is being mounted GROUP, then it
667      1195      2  ! cannot be mounted SHARE, or SYSTEM.
668      1196      2  !
669      1197      2  IF .MOUNT_OPTIONS [OPT_GROUP]
670      1198      2  AND (.MOUNT_OPTIONS [OPT_SHARE] OR .MOUNT_OPTIONS [OPT_SYSTEM])
671      1199      2  THEN
672      1200      2  RETURN (MOUN$_CONFQUAL);
```

```

673 1201 2
674 1202 2
675 1203 2 If the volume is being mounted SYSTEM, then it
676 1204 2 cannot be mounted SHARE, or GROUP.
677 1205 2
678 1206 2 IF .MOUNT_OPTIONS [OPT_SYSTEM]
679 1207 2 AND (.MOUNT_OPTIONS [OPT_SHARE] OR .MOUNT_OPTIONS [OPT_GROUP])
680 1208 2 THEN
681 1209 2 RETURN (MOUN$_CONFQUAL);
682 1210 2
683 1211 2
684 1212 2 If the volume is being mounted SHARE then it
685 1213 2 cannot be mounted SYSTEM or GROUP.
686 1214 2
687 1215 2 IF .MOUNT_OPTIONS [OPT_SHARE]
688 1216 2 AND (.MOUNT_OPTIONS [OPT_SYSTEM] OR .MOUNT_OPTIONS [OPT_GROUP])
689 1217 2 THEN
690 1218 2 RETURN (MOUN$_CONFQUAL);
691 1219 2
692 1220 2
693 1221 2 Do not allow user to override volume identification if requesting
694 1222 2 a SYSTEM, GROUP or SHARE mount.
695 1223 2
696 1224 2 IF .MOUNT_OPTIONS [OPT_OVR_ID]
697 1225 2 AND (.MOUNT_OPTIONS [OPT_SYSTEM] OR
698 1226 2 .MOUNT_OPTIONS [OPT_GROUP] OR
699 1227 2 .MOUNT_OPTIONS [OPT_SHARE])
700 1228 2 THEN
701 1229 2 RETURN (MOUN$_CONFQUAL);
702 1230 2
703 1231 2
704 1232 2 If no device names have been seen,
705 1233 2 reject the mount request.
706 1234 2
707 1235 2 IF NOT .MOUNT_OPTIONS [OPT_DEVICE]
708 1236 2 THEN
709 1237 2 RETURN (SS$_BADPARAM);
710 1238 2
711 1239 2
712 1240 2 If no volume label seen, make sure the user has
713 1241 2 done this on purpose. If not, exit.
714 1242 2
715 1243 2 IF NOT .MOUNT_OPTIONS [OPT_LABEL] ! Indicate volume label seen
716 1244 2 AND NOT (.MOUNT_OPTIONS [OPT_OVR_ID] OR
717 1245 2 .MOUNT_OPTIONS [OPT_FOREIGN])
718 1246 2 THEN
719 1247 2 RETURN (SS$_BADPARAM);
720 1248 2
721 1249 2
722 1250 2 If the device is being mounted /FOREIGN, then many of
723 1251 2 the qualifiers having to do with FILES-11 are not legal.
724 1252 2
725 1253 2 IF .MOUNT_OPTIONS [OPT_FOREIGN]
726 1254 2 AND (.MOUNT_OPTIONS [OPT_ACCESSED] OR
727 1255 2 .MOUNT_OPTIONS [OPT_UNIQUEACP] OR
728 1256 2 .MOUNT_OPTIONS [OPT_FILEACP] OR
729 1257 2 .MOUNT_OPTIONS [OPT_SAMEACP] OR
```



```

730 1258 3 .MOUNT_OPTIONS [OPT_BIND] OR
731 1259 3 .MOUNT_OPTIONS [OPT_CACHE] OR
732 1260 3 .MOUNT_OPTIONS [OPT_WINDOW] OR
733 1261 3 .MOUNT_OPTIONS [OPT_EXTENSION]
734 1262 3 )
735 1263 3 THEN
736 1264 3 RETURN (MOUN$_CONFQUAL);
737 1265 3
738 1266 3 !+
739 1267 3 ! Perform some preliminary privilege checks.
740 1268 3 !-
741 1269 3
742 1270 3 PHD = .CTL$GL_PHD; ! Get the process header address
743 1271 3
744 1272 3 IF .MOUNT_OPTIONS [OPT_GROUP] ! If /GROUP, user must have GRPNAM
745 1273 3 AND NOT .BBLOCK [PHD [PHD$_PRIVMSK], PRV$_GRPNAM]
746 1274 3 THEN
747 1275 3 RETURN (SS$_NOGRPNAM);
748 1276 3
749 1277 3 IF .MOUNT_OPTIONS [OPT_SYSTEM] ! If /SYSTEM, user must have SYSNAM
750 1278 3 AND NOT .BBLOCK [PHD [PHD$_PRIVMSK], PRV$_SYSNAM]
751 1279 3 THEN
752 1280 3 RETURN (SS$_NOSYSNAM);
753 1281 3
754 1282 3 IF (.MOUNT_OPTIONS [OPT_UNIQUEACP] OR ! This check must be performed AFTER the /PROCESSOR
755 1283 3 .MOUNT_OPTIONS [OPT_SAMEACP] OR ! qualifier has been processed.
756 1284 3 .MOUNT_OPTIONS [OPT_FILEACP])
757 1285 3 AND NOT .BBLOCK [PHD [PHD$_PRIVMSK], PRV$_OPER]
758 1286 3 THEN
759 1287 3 RETURN (SS$_NOOPER);
760 1288 3
761 1289 3 IF .MOUNT_OPTIONS [OPT_FILEACP] ! Must have CMK to use
762 1290 3 AND NOT .BBLOCK [PHD [PHD$_PRIVMSK], PRV$_CHKRNL] ! a special ACP
763 1291 3 THEN
764 1292 3 RETURN (SS$_NOCMKRNL);
765 1293 3
766 1294 3
767 1295 3 !++
768 1296 3 ! If we get this far, the parameters have passed the
769 1297 3 ! preliminary checks. Return a successful status.
770 1298 3 !--
771 1299 3
772 1300 3 SS$_NORMAL
773 1301 3
774 1302 3 1 END; ! End of routine CHECK_PARAMETERS
```

```

.TITLE MOUPAR
.IDENT \V04-000\
.PSECT SOWNS,NOEXE,2
```

```

00000 OWN_START:
       .BLKB 0
00000 DEVICE_BUFFER:
       .BLKB 1008
003F0 LABEL_BUFFER:
```

J 12
16-Sep-1984 01:22:31
14-Sep-1984 12:45:31VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[MOUNT.SRC]MOUPAR.B32;1 Page 16
(4)

```
007E0 LOG_NAME .BLKB 1008
          .BUFFER:
0081F          .BLKB 63
00820 ACP_NAME .BLKB 1
          .BUFFER:
0085F          .BLKB 63
00860 VOLSET .BLKB 1
          .BUFFER:
0089F          .BLKB 63
008A0 COMMENT .BLKB 1
          .BUFFER:
008F0 TPARSE_BLOCK:
          .BLKB 80
00914 OWN_END: .BLKB 36
          .BLKB 0

          .PSECT $GLOBAL$,NOEXE,2

00000 GLOBAL_START::
          .BLKB 0
00000 MOUNT_OPTIONS::
          .BLKB 8
00008 MOUNT_FLAGS::
          .BLKB 4
0000C PROTECTION::
          .BLKB 4
00010 OWNER_UIC::
          .BLKB 4
00014 USER_UIC::
          .BLKB 4
00018 EXTENSION::
          .BLKB 4
0001C WINDOW::
          .BLKB 4
00020 ACCESSED::
          .BLKB 4
00024 BLOCKSIZE::
          .BLKB 4
00028 RECORDSZ::
          .BLKB 4
0002C FID_CACHE::
          .BLKB 4
00030 EXT_CACHE::
          .BLKB 4
00034 QUO_CACHE::
          .BLKB 4
00038 EXT_LIMIT::
          .BLKB 4
0003C DEVICE_COUNT::
          .BLKB 4
00040 LABEL_COUNT::
          .BLKB 4
00044 VID_STRING::
          .BLKB 8
0004C COMMENT_STRING::
          .BLKB 8
00054 ACP_STRING::
          .BLKB 8
```


0005C LOG_NAME::
 .BKLB 8
00064 STRUCT_NAME::
 .BKLB 8
0006C DRIVE_COUNT::
 .BKLB 64
000AC DEVICE_STRING::
 .BKLB 128
0012C LABEL_STRING::
 .BKLB 128
001AC JRNL_SIZE::
 .BKLB 4
001B0 JRNL_EXTEND::
 .BKLB 4
001B4 JRNL_QUOTA::
 .BKLB 4
001B8 JRNL_RECORD_SIZE::
 .BKLB 4
001BC GLOBAL_END::
 .BKLB 0

.EXTRN DEVCHAR_DESC, DEVICE_CHAR
.EXTRN NEWLINE, CTL\$GL_PHD
.EXTRN LIB\$TPARSE

.PSECT \$CODE\$,NOWRT,2

OFFC 00000

.ENTRY CHECK_PARAMS, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 0762
R10,R11
MOVAB GLOBAL_START, R11
SUBL2 #4, SP
MOVCS #0, (SP), #0, #444, GLOBAL_START 0825
MOVCS #0, (SP), #0, #2324, OWN_START 0826
MOVL #8, TPARSE_BLOCK 0831
MOVL #3, TPARSE_BLOCK+4 0832
CLRQ MOUNT_OPTIONS+4 0850
CLRL MOUNT_OPTIONS
MOVQ #1, STATUS 0851
CLRQ DEVICE_COUNT 0853
MOVL ITEM_LIST, ITEM 0855
PROBER #0, #4, (ITEM) 0856
BNEQ 1\$
MOVL #12, R0 0858
RET
BLBS STATUS, 2\$ 0860
BRW 62\$
TSTW 2(ITEM)
BNEQ 3\$
BRW 57\$
PROBER #0, #12, (ITEM) 0871
BNEQ 4\$
MOVL #12, STATUS 0873
BRB 1\$
MOVZWL 2(ITEM), R2 0876
CMPW R2, #1 0881

01BC	8F	00	5B	0000'	CF	9E	00002	
			5E		04	C2	00007	
			6E		00	2C	0000A	
					6B		00011	
0914	8F	00	6E	0000'	00	2C	00012	
				0000'	CF		00019	
				CF	08	D0	0001C	
				CF	03	D0	00021	
					04	AB	7C	00026
					6B	D4	00029	
			58		01	7D	0002B	
				3C	AB	7C	0002E	
			56	04	AC	D0	00031	
		66	04		00	0C	00035	
					04	12	00039	
			50		0C	D0	0003B	
						04	0003E	
			03		58	E8	0003F	1\$:
					04CD	31	00042	
				02	A6	B5	00045	2\$:
					03	12	00048	
					032B	31	0004A	
		66	0C		00	0C	0004D	3\$:
					05	12	00051	
			58		0C	D0	00053	
					E7	11	00056	
			52	02	A6	3C	00058	4\$:
			01		52	B1	0005C	

			42	12	0005F	BNEQ	7\$		
		10	3C	AB	D1	00061	CMPL	DEVICE_COUNT, #16	
				33	18	00065	BGEQ	6\$	
	03	AB	40	8F	88	00067	BISB2	#64, MOUNT_OPTIONS+3	0884
			3C	AB	D6	0006C	INCL	DEVICE_COUNT	0885
50	3C	AB		01	78	0006F	ASHL	#1, DEVICE_COUNT, R0	0886
		51		66	3C	00074	MOVZWL	(ITEM), R1	
		3F		51	B1	00077	CMPW	R1, #63	
				03	1B	0007A	BLEQU	5\$	
		51		3F	D0	0007C	MOVL	#63, R1	
	00A4	CB40		51	D0	0007F	MOVL	R1, DEVICE_STRING-8[R0]	
51	3C	AB		3F	C5	00085	MULL3	#63, DEVICE_COUNT, R1	0887
	00A8	CB40	0000'	CF41	9E	0008A	MOVAB	DEVICE_BUFFER[R1], DEVICE_STRING-4[R0]	
			00A8	CB40	DD	00093	PUSHL	DEVICE_STRING-4[R0]	0888
				45	11	00098	BRB	9\$	
	58	00728084		8F	D0	0009A	MOVL	#7504004, STATUS	0881
				45	11	000A1	BRB	11\$	
	02			52	B1	000A3	CMPW	R2, #2	0893
				42	12	000A6	BNEQ	12\$	
	10		40	AB	D1	000A8	CMPL	LABEL_COUNT, #16	
				33	18	000AC	BGEQ	10\$	
	03	AB	80	8F	88	000AE	BISB2	#128, MOUNT_OPTIONS+3	0896
			40	AB	D6	000B3	INCL	LABEL_COUNT	0897
50	40	AB		01	78	000B6	ASHL	#1, LABEL_COUNT, R0	0898
		51		66	3C	000BB	MOVZWL	(ITEM), RT	
		3F		51	B1	000BE	CMPW	R1, #63	
				03	1B	000C1	BLEQU	8\$	
		51		3F	D0	000C3	MOVL	#63, R1	
	0124	CB40		51	D0	000C6	MOVL	R1, LABEL_STRING-8[R0]	
51	40	AB		3F	C5	000CC	MULL3	#63, LABEL_COUNT, R1	0899
	0128	CB40	0000'	CF41	9E	000D1	MOVAB	LABEL_BUFFER[R1], LABEL_STRING-4[R0]	
			0128	CB40	DD	000DA	PUSHL	LABEL_STRING-4[R0]	0900
				6D	11	000DF	BRB	18\$	
	58	0072808C		8F	D0	000E1	MOVL	#7504012, STATUS	0893
				43	11	000E8	BRB	15\$	
	03			52	B1	000EA	CMPW	R2, #3	0905
				41	12	000ED	BNEQ	16\$	
	03	AB		20	88	000EF	BISB2	#32, MOUNT_OPTIONS+3	0906
		50		66	3C	000F3	MOVZWL	(ITEM), R0	0907
		3F		50	B1	000F6	CMPW	R0, #63	
				03	1B	000F9	BLEQU	13\$	
	50			3F	D0	000FB	MOVL	#63, R0	
	5C	AB		50	D0	000FE	MOVL	R0, LOG_NAME	
	60	AB	0000'	CF	9E	00102	MOVAB	LOG_NAME_BUFFER, LOG_NAME+4	0908
			0000'	CF	9F	00103	PUSHAB	LOG_NAME_BUFFER	0909
				3F	DD	0010C	PUSHL	#63	
				56	DD	0010E	PUSHL	ITEM	
	0000V	CF		03	FB	00110	CALLS	#3, COPY ITEM	
		58		50	D0	00115	MOVL	R0, STATUS	
60	BB	5C		3A	3A	00118	LOCC	#58, LOG_NAME, @LOG_NAME+4	0914
				02	12	0011E	BNEQ	14\$	
				51	D4	00120	CLRL	R1	
		5A		51	D0	00122	MOVL	R1, PTR	
				06	13	00125	BEQL	15\$	0915
5C	AB	5A	60	AB	C3	00127	SUBL3	LOG_NAME+4, PTR, LOG_NAME	0917
			0242	31	0012D	BRW	56\$		0918
		06		52	B1	00130	CMPW	R2, #6	0921

	50		1B	12	00133	BNEQ	19\$		
	3F		66	3C	00135	MOVZWL	(ITEM), R0		0922
			50	B1	00138	CMPW	R0, #63		
	50		03	1B	0013B	BLEQU	17\$		
	AB		3F	D0	0013D	MOVL	#63, R0		
54			50	D0	00140	MOVL	R0, ACP_STRING		
58	AB	0000'	CF	9E	00144	MOVAB	ACP_NAME_BUFFER, ACP_STRING+4		0923
		0000'	CF	9F	0014A	PUSHAB	ACP_NAME_BUFFER		0924
			22	11	0014E	BRB	21\$		
	07		52	B1	00150	CMPW	R2, #7		0927
			22	12	00153	BNEQ	22\$		
	50		66	3C	00155	MOVZWL	(ITEM), R0		0928
	3F		50	B1	00158	CMPW	R0, #63		
			03	1B	0015B	BLEQU	20\$		
	50		3F	D0	0015D	MOVL	#63, R0		
64	AB		50	D0	00160	MOVL	R0, STRUCT_NAME		
68	AB	0000'	CF	9E	00164	MOVAB	VOLSET_BUFFER, STRUCT_NAME+4		0929
05	AB		01	88	0016A	BISB2	#1, MOUNT_OPTIONS+5		0930
		0000'	CF	9F	0016E	PUSHAB	VOLSET_BUFFER		0931
			3F	DD	00172	PUSHL	#63		
			01	EC	31	BRW	54\$		
	04		52	B1	00177	CMPW	R2, #4		0938
			05	12	0017A	BNEQ	23\$		
		08	AB	9F	0017C	PUSHAB	MOUNT_FLAGS		
			0C	11	0017F	BRB	24\$		
	05		52	B1	00181	CMPW	R2, #5		0940
			0A	12	00184	BNEQ	25\$		
03	AB		02	88	00186	BISB2	#2, MOUNT_OPTIONS+3		0941
		20	AB	9F	0018A	PUSHAB	ACCESSED		0942
			01	D1	31	BRW	53\$		
	08		52	B1	00190	CMPW	R2, #8		0945
			21	12	00193	BNEQ	26\$		
02	AB		01	88	00195	BISB2	#1, MOUNT_OPTIONS+2		0946
		24	AB	9F	00199	PUSHAB	BLOCKSIZE		0947
			04	DD	0019C	PUSHL	#4		
			56	DD	0019E	PUSHL	ITEM		
0000V	CF		03	FB	001A0	CALLS	#3, COPY_ITEM		
	58		50	D0	001A5	MOVL	R0, STATUS		
	82		58	E9	001AB	BLBC	STATUS, 15\$		
0000FFE	8F	24	AB	D1	001AB	CMPL	BLOCKSIZE, #65534		0949
			01	11	31	BRW	40\$		
	09		52	B1	001B6	CMPW	R2, #9		0955
			4D	12	001B9	BNEQ	29\$		
	6B		01	88	001BB	BISB2	#1, MOUNT_OPTIONS		0956
	6B		02	8A	001BE	BICB2	#2, MOUNT_OPTIONS		0957
05	AB		08	8A	001C1	BICB2	#8, MOUNT_OPTIONS+5		0958
			5E	DD	001C5	PUSHL	SP		0959
			04	DD	001C7	PUSHL	#4		
			56	DD	001C9	PUSHL	ITEM		
0000V	CF		03	FB	001CB	CALLS	#3, COPY_ITEM		
	58		50	D0	001D0	MOVL	R0, STATUS		
	7B		58	E9	001D3	BLBC	STATUS, 31\$		
00000320	50		6E	D0	001D6	MOVL	DENSITY, R0		0961
	8F		50	D1	001D9	CMPL	R0, #800		0963
			05	12	001E0	BNEQ	27\$		
	6B		02	88	001E2	BISB2	#2, MOUNT_OPTIONS		
			6A	11	001E5	BRB	31\$		

00000640	8F		50	D1	001E7	27\$:	CMPL	R0	#1600	:	0964
			06	12	001EE		BNEQ	28\$		:	
05	AB		08	88	001F0		BISB2	#8	MOUNT_OPTIONS+5	:	
			58	11	001F4		BRB	31\$		:	
0000186A	8F		50	D1	001F6	28\$:	CMPL	R0	#6250	:	0965
			52	13	001FD		BEQL	31\$		:	
	58	00728014	8F	D0	001FF		MOVL	#7503892	STATUS	:	0966
			49	11	00206		BRB	31\$		:	0961
	0A		52	B1	00208	29\$:	CMPW	R2	#10	:	0971
			20	12	0020B		BNEQ	30\$		:	
		30	AB	9F	0020D		PUSHAB	EXT_CACHE		:	0972
			04	DD	00210		PUSHL	#4		:	
			56	DD	00212		PUSHL	ITEM		:	
0000V	CF		03	FB	00214		CALLS	#3	COPY_ITEM	:	
	58		50	D0	00219		MOVL	R0	STATUS	:	
05	AB	A0	8F	8A	0021C		BICB2	#160	MOUNT_OPTIONS+5	:	0974
		30	AB	D5	00221		TSTL	EXT_CACHE		:	0975
			78	14	00224		BGTR	37\$		:	
05	AB	80	8F	88	00226		BISB2	#128	MOUNT_OPTIONS+5	:	0977
			75	11	0022B		BRB	38\$		:	
	0B		52	B1	0022D	30\$:	CMPW	R2	#11	:	0983
			21	12	00230		BNEQ	32\$		:	
		2C	AB	9F	00232		PUSHAB	FID_CACHE		:	0984
			04	DD	00235		PUSHL	#4		:	
			56	DD	00237		PUSHL	ITEM		:	
0000V	CF		03	FB	00239		CALLS	#3	COPY_ITEM	:	
	58		50	D0	0023E		MOVL	R0	STATUS	:	
05	AB	0120	8F	AA	00241		BICW2	#288	MOUNT_OPTIONS+5	:	0985
	01	2C	AB	D1	00247		CMPL	FID_CACHE	#1	:	0987
			51	14	0024B		BGTR	37\$		:	
06	AB		01	88	0024D		BISB2	#1	MOUNT_OPTIONS+6	:	0989
			7D	11	00251	31\$:	BRB	41\$		:	
	0C		52	B1	00253	32\$:	CMPW	R2	#12	:	0995
			05	12	00256		BNEQ	33\$		:	
		38	AB	9F	00258		PUSHAB	EXT_LIMIT		:	
			0C	11	0025B		BRB	34\$		:	
	0D		52	B1	0025D	33\$:	CMPW	R2	#13	:	0997
			09	12	00260		BNEQ	35\$		:	
02	AB		04	88	00262		BISB2	#4	MOUNT_OPTIONS+2	:	0998
		10	AB	9F	00266		PUSHAB	OWNER_UIC		:	0999
			74	11	00269	34\$:	BRB	43\$		:	
	0E		52	B1	0026B	35\$:	CMPW	R2	#14	:	1002
			09	12	0026E		BNEQ	36\$		:	
02	AB		02	88	00270		BISB2	#2	MOUNT_OPTIONS+2	:	1003
		0C	AB	9F	00274		PUSHAB	PROTECTION		:	1004
			75	11	00277		BRB	45\$		:	
	0F		52	B1	00279	36\$:	CMPW	R2	#15	:	1007
			26	12	0027C		BNEQ	39\$		:	
		34	AB	9F	0027E		PUSHAB	QUO_CACHE		:	1008
			04	DD	00281		PUSHL	#4		:	
			56	DD	00283		PUSHL	ITEM		:	
0000V	CF		03	FB	00285		CALLS	#3	COPY_ITEM	:	
	58		50	D0	0028A		MOVL	R0	STATUS	:	
05	AB	0220	8F	AA	0028D		BICW2	#544	MOUNT_OPTIONS+5	:	1009
		34	AB	D5	00293		TSTL	QUO_CACHE		:	1011
			06	14	00296		BGTR	37\$		:	
06	AB		02	88	00298		BISB2	#2	MOUNT_OPTIONS+6	:	1013

05	AB		32	11	0029C	BRB	41\$		
			20	88	0029E	BISB2	#32, MOUNT_OPTIONS+5	1015	
			2C	11	002A2	BRB	41\$	1016	
	10		52	B1	002A4	CMPW	R2, #16	1019	
			2A	12	002A7	BNEQ	42\$		
04	AB		20	88	002A9	BISB2	#32, MOUNT_OPTIONS+4	1020	
		28	AB	9F	002AD	PUSHAB	RECORDSZ	1021	
			04	DD	002B0	PUSHL	#4		
			56	DD	002B2	PUSHL	ITEM		
0000V	CF		03	FB	002B4	CALLS	#3, COPY_ITEM		
	58		50	D0	002B9	MOVL	R0, STATUS		
	11		58	E9	002BC	BLBC	STATUS, 41\$		
0000FFFE	8F	28	AB	D1	002BF	CMPL	RECORDSZ, #65534	1023	
			07	15	002C7	BLEQ	41\$		
	58	0072817C	8F	D0	002C9	MOVL	#7504252, STATUS	1025	
			009F	31	002D0	BRW	56\$	1026	
	11		52	B1	002D3	CMPW	R2, #17	1029	
			09	12	002D6	BNEQ	44\$		
03	AB		01	88	002D8	BISB2	#1, MOUNT_OPTIONS+3	1030	
		1C	AB	9F	002DC	PUSHAB	WINDOW	1031	
			72	11	002DF	BRB	51\$		
	12		52	B1	002E1	CMPW	R2, #18	1034	
			0A	12	002E4	BNEQ	46\$		
02	AB		8F	88	002E6	BISB2	#128, MOUNT_OPTIONS+2	1035	
		80	AB	9F	002EB	PUSHAB	EXTENSION	1036	
		18	71	11	002EE	BRB	53\$		
	14		52	B1	002F0	CMPW	R2, #20	1039	
			36	12	002F3	BNEQ	48\$		
	6B		08	88	002F5	BISB2	#8, MOUNT_OPTIONS	1043	
	51		66	3C	002F8	MOVZWL	(ITEM), RT	1044	
0050	8F		51	B1	002FB	CMPW	R1, #80		
			04	1B	00300	BLEQU	47\$		
	51	50	8F	9A	00302	MOVZBL	#80, R1		
	57	0000G	CF	3C	00306	MOVZWL	NEWLINE, R7		
	51		57	C1	0030B	ADDL3	R7, R1, COMMENT_STRING		
4C	AB		57	9E	00310	MOVAB	COMMENT_BUFFER, COMMENT_STRING+4	1045	
0000'	CF	50	57	28	00316	MOVC3	R7, BNEQLINE+4, COMMENT_BUFFER	1046	
		0000G	47	9F	0031E	PUSHAB	COMMENT_BUFFER[R7]	1049	
			A7	9F	00323	PUSHAB	-80(R7)	1048	
	6E		6E	CE	00326	MNEGL	(SP), (SP)		
			38	11	00329	BRB	54\$	1047	
	15		52	B1	0032B	CMPW	R2, #21	1053	
			09	12	0032E	BNEQ	49\$		
	59		01	D0	00330	MOVL	#1, JRNL_ITEM_SEEN	1054	
		01AC	CB	9F	00333	PUSHAB	JRNL_SIZE	1055	
			28	11	00337	BRB	53\$		
	16		52	B1	00339	CMPW	R2, #22	1058	
			09	12	0033C	BNEQ	50\$		
	59		01	D0	0033E	MOVL	#1, JRNL_ITEM_SEEN	1059	
		01B0	CB	9F	00341	PUSHAB	JRNL_EXTEND	1060	
			1A	11	00345	BRB	53\$		
	17		52	B1	00347	CMPW	R2, #23	1063	
			09	12	0034A	BNEQ	52\$		
	59		01	D0	0034C	MOVL	#1, JRNL_ITEM_SEEN	1064	
		01B4	CB	9F	0034F	PUSHAB	JRNL_QUOTA	1065	
			0C	11	00353	BRB	53\$		
	18		52	B1	00355	CMPW	R2, #24	1068	

				59		15	12	00358	BNEQ	55\$		
						01	00	0035A	MOVL	#1, JRNL_ITEM_SEEN	1069	
					01B8	CB	9F	0035D	PUSHAB	JRNL_RECORD_SIZE	1070	
						04	DD	00361	PUSHL	#4		
						56	DD	00363	PUSHL	ITEM		
				0000V	CF	03	FB	00365	CALLS	#3, COPY_ITEM		
				58		50	DO	0036A	MOVL	R0, STATUS		
						03	11	0036D	BRB	56\$		
				58		14	DO	0036F	MOVL	#20, STATUS	1073	
				56		0C	CO	00372	ADDL2	#12, ITEM	1076	
						FCC7	31	00375	BRW	1\$	0860	
				03		58	E8	00378	BLBS	STATUS, 58\$	1083	
						0194	31	0037B	BRW	62\$		
				01		04	EF	0037E	EXTZV	#4, #1, MOUNT_FLAGS+3, R0	1099	
07	50	0B	AB	06		50	FO	00384	INSV	R0, #6, #1, MOUNT_OPTIONS+7		
	AB	09	AB	01		06	EF	0038A	EXTZV	#6, #1, MOUNT_FLAGS+1, R0	1100	
01	50		01	00		50	FO	00390	INSV	R0, #0, #1, MOUNT_OPTIONS+1		
	AB	08	AB	01		01	EF	00396	EXTZV	#1, #1, MOUNT_FLAGS, R0	1101	
	50		01	07		50	FO	0039C	INSV	R0, #7, #1, MOUNT_OPTIONS		
	6B	0A	AB	01		03	EF	003A1	EXTZV	#3, #1, MOUNT_FLAGS+2, R0	1102	
	50			50		50	D2	003A7	MCOML	R0, R0		
06	AB		01	06		50	FO	003AA	INSV	R0, #6, #1, MOUNT_OPTIONS+6		
	50	08	AB	01		03	EF	003B0	EXTZV	#3, #1, MOUNT_FLAGS, R0	1103	
05	AB		01	02		50	FO	003B6	INSV	R0, #2, #1, MOUNT_OPTIONS+5		
	50	08	AB	01		04	EF	003BC	EXTZV	#4, #1, MOUNT_FLAGS, R0	1104	
05	AB		01	04		50	FO	003C2	INSV	R0, #4, #1, MOUNT_OPTIONS+5		
	50	09	AB	01		04	EF	003C8	EXTZV	#4, #1, MOUNT_FLAGS+1, R0	1105	
	6B		01	06		50	FO	003CE	INSV	R0, #6, #1, MOUNT_OPTIONS		
	50	08	AB	01		06	EF	003D3	EXTZV	#6, #1, MOUNT_FLAGS, R0	1106	
				50		50	D2	003D9	MCOML	R0, R0		
01	AB		01	01		50	FO	003DC	INSV	R0, #1, #1, MOUNT_OPTIONS+1		
	50	0A	AB	01		01	EF	003E2	EXTZV	#1, #1, MOUNT_FLAGS+2, R0	1107	
06	AB		01	04		50	FO	003E8	INSV	R0, #4, #1, MOUNT_OPTIONS+6		
	50	0A	AB	01		02	EF	003EE	EXTZV	#2, #1, MOUNT_FLAGS+2, R0	1108	
06	AB		01	05		50	FO	003F4	INSV	R0, #5, #1, MOUNT_OPTIONS+6		
	50	09	AB	01		05	EF	003FA	EXTZV	#5, #1, MOUNT_FLAGS+1, R0	1109	
06	AB		01	03		50	FO	00400	INSV	R0, #3, #1, MOUNT_OPTIONS+6		
	50	09	AB	01		03	EF	00406	EXTZV	#3, #1, MOUNT_FLAGS+1, R0	1110	
04	AB		01	03		50	FO	0040C	INSV	R0, #3, #1, MOUNT_OPTIONS+4		
	50	09	AB	01		07	EF	00412	EXTZV	#7, #1, MOUNT_FLAGS+1, R0	1111	
04	AB		01	04		50	FO	00418	INSV	R0, #4, #1, MOUNT_OPTIONS+4		
05	AB		01	06		0A	AB	FO 0041E	INSV	MOUNT_FLAGS+2, #6, #1, MOUNT_OPTIONS+5	1112	
	50	08	AB	01		02	EF	00425	EXTZV	#2, #1, MOUNT_FLAGS, R0	1113	
				50		50	D2	0042B	MCOML	R0, R0		
06	AB		01	02		50	FO	0042E	INSV	R0, #2, #1, MOUNT_OPTIONS+6		
01	AB		01	03		08	AB	FO 00434	INSV	MOUNT_FLAGS, #3, #1, MOUNT_OPTIONS+1	1114	
02	AB		01	04		09	AB	FO 0043B	INSV	MOUNT_FLAGS+1, #4, #1, MOUNT_OPTIONS+2	1115	
	50	09	AB	01		01	EF	00442	EXTZV	#1, #1, MOUNT_FLAGS+1, R0	1116	
02	AB		01	06		50	FO	00448	INSV	R0, #6, #1, MOUNT_OPTIONS+2		
	50	09	AB	01		02	EF	0044E	EXTZV	#2, #1, MOUNT_FLAGS+1, R0	1117	
02	AB		01	05		50	FO	00454	INSV	R0, #5, #1, MOUNT_OPTIONS+2		
	50	08	AB	01		07	EF	0045A	EXTZV	#7, #1, MOUNT_FLAGS, R0	1118	
04	AB		01	06		50	FO	00460	INSV	R0, #6, #1, MOUNT_OPTIONS+4		
				01		8F	88	00466	BISB2	#128, MOUNT_OPTIONS+1	1119	
	50	0A	AB	01		04	EF	0046B	EXTZV	#4, #1, MOUNT_FLAGS+2, R0	1120	
01	AB		01	02		50	FO	00471	INSV	R0, #2, #1, MOUNT_OPTIONS+1		
	50	0A	AB	01		05	EF	00477	EXTZV	#5, #1, MOUNT_FLAGS+2, R0	1121	

06	AB		01	07	50	FO	0047D	INSV	R0, #7, #1, MOUNT_OPTIONS+6	:	
	50	0A	AB	01	06	EF	004E3	EXTZV	#6, #1, MOUNT_FLAGS+2, R0	:	1122
07	AB		01	00	50	FO	004B9	INSV	R0, #0, #1, MOUNT_OPTIONS+7	:	
	50	0A	AB	01	07	EF	0048F	EXTZV	#7, #1, MOUNT_FLAGS+2, R0	:	1123
07	AB		01	01	50	FO	00495	INSV	R0, #1, #1, MOUNT_OPTIONS+7	:	
07	AB		01	02	0B	AB	FO 0049B	INSV	MOUNT_FLAGS+3, #2, #1, MOUNT_OPTIONS+7	:	1124
	50	0B	AB	01	01	EF	004A2	EXTZV	#1, #1, MOUNT_FLAGS+3, R0	:	1125
07	AB		01	03	50	FO	004A8	INSV	R0, #3, #1, MOUNT_OPTIONS+7	:	
	50	0B	AB	01	02	EF	004AE	EXTZV	#2, #1, MOUNT_FLAGS+3, R0	:	1126
07	AB		01	04	50	FO	004B4	INSV	R0, #4, #1, MOUNT_OPTIONS+7	:	
	50	0B	AB	01	03	EF	004BA	EXTZV	#3, #1, MOUNT_FLAGS+3, R0	:	1127
07	AB		01	05	50	FO	004C0	INSV	R0, #5, #1, MOUNT_OPTIONS+7	:	
	50	0B	AB	01	05	EF	004C6	EXTZV	#5, #1, MOUNT_FLAGS+3, R0	:	1128
07	AB		01	07	50	FO	004CC	INSV	R0, #7, #1, MOUNT_OPTIONS+7	:	
	AB		01	08	07	AB	E9 004D2	BLBC	MOUNT_OPTIONS+7, 60\$	:	1139
					06	AB	95 004D6	TSTB	MOUNT_OPTIONS+6	:	
						03	18 004D9	BGEQ	60\$	:	
						00E3	31 004DB	BRW	75\$	:	
				04	59	E9	004DE	BLBC	JRNL_ITEM SEEN, 61\$	:	1140
				F6	07	AB	E9 004E1	BLBC	MOUNT_OPTIONS+7, 59\$	:	
					54	AB	D5 004E5	TSTL	ACP_STRING	:	1148
						3F	13 004E8	BEQL	65\$	:	
				52	54	AB	9E 004EA	MOVAB	ACP_STRING, STR_DESCRIPTOR	:	1155
				CF		62	3C 004EE	MOVZWL	(STR_DESCRIPTOR), TPARSE_BLOCK+8	:	1156
				0000'		A2	D0 004F3	MOVL	4(STR_DESCRIPTOR), TPARSE_BLOCK+12	:	1157
				0000'		CF	9F 004F9	PUSHAB	ACP_KTB	:	1158
						04	CF 9F 004FD	PUSHAB	ACP_STB	:	
						0000V	CF 9F 00501	PUSHAB	TPARSE_BLOCK	:	
						0000V	03 FB 00505	CALLS	#3, LIB\$TPARSE	:	
						0000'	50 D0 0050C	MOVL	R0, STATUS	:	
							58 E8 0050F	BLBS	STATUS, 63\$	:	
							58 D0 00512	MOVL	STATUS, R0	:	1160
							04 00515	RET		:	
							03 E0 00516	BBS	#3, MOUNT_OPTIONS+3, 64\$	:	1164
							04 E1 0051B	BBC	#4, MOUNT_OPTIONS+3, 65\$	:	1165
							62 3C 00520	MOVZWL	(STR_DESCRIPTOR), ACP_STRING	:	1168
							A2 D0 00524	MOVL	4(STR_DESCRIPTOR), ACP_STRING+4	:	1169
							06 E1 00529	BBC	#6, MOUNT_OPTIONS+7, 66\$	:	1178
							6B 95 0052E	TSTB	MOUNT_OPTIONS	:	1179
							04 19 00530	BLSS	66\$	:	
							01 88 00532	BISB2	#1, MOUNT_OPTIONS+1	:	1181
							06 E0 00536	BBS	#6, MOUNT_OPTIONS, 67\$	:	1187
							6B 95 0053A	TSTB	MOUNT_OPTIONS	:	1188
							07 19 0053C	BLSS	67\$	:	
							01 AB E8 0053E	BLBS	MOUNT_OPTIONS+1, 67\$	:	1189
							10 88 00542	BISB2	#16, MOUNT_OPTIONS	:	1191
							07 EF 00545	EXTZV	#7, #1, MOUNT_OPTIONS, R2	:	1197
							52 EF 0054A	BLBC	R2, 68\$	:	
							06 E0 0054D	BBS	#6, MOUNT_OPTIONS, 75\$	:	1198
							01 AB E8 00551	BLBS	MOUNT_OPTIONS+1, 75\$	:	
							00 EF 00555	EXTZV	#0, #1, MOUNT_OPTIONS+1, R1	:	1206
							51 EF 0055B	BLBC	R1, 69\$	:	
							06 E0 0055E	BBS	#6, MOUNT_OPTIONS, 75\$	:	1207
							52 EF 00562	BLBS	R2, 75\$	:	
							06 E1 00565	BBC	#6, MOUNT_OPTIONS, 70\$	:	1215
							51 E8 00569	BLBS	R1, 75\$	:	1216
							52 E8 0056C	BLBS	R2, 75\$	:	

0A	02	AB	06	E1	0056F	70\$:	BBC	#6, MOUNT_OPTIONS+2, 71\$	: 1224
		4A	51	E8	00574		BLBS	R1, 75\$	: 1225
		47	52	E8	00577		BLBS	R2, 75\$	: 1226
43		6B	06	E0	0057A		BBS	#6, MOUNT_OPTIONS, 75\$	: 1227
0F	03	AB	06	E1	0057E	71\$:	BBC	#6, MOUNT_OPTIONS+3, 72\$	: 1235
			AB	95	00583		TSTB	MOUNT_OPTIONS+3	: 1243
			0E	19	00586		BLSS	73\$	: 1244
09	02	AB	06	E0	00588		BBS	#6, MOUNT_OPTIONS+2, 73\$	: 1245
09	01	AB	03	E0	0058D		BBS	#3, MOUNT_OPTIONS+1, 74\$	: 1247
		50	14	D0	00592	72\$:	MOVL	#20, R0	: 1253
				04	00595		RET		: 1254
2E	01	AB	03	E1	00596	73\$:	BBC	#3, MOUNT_OPTIONS+1, 76\$	: 1255
21	03	AB	01	E0	0059B	74\$:	BBS	#1, MOUNT_OPTIONS+3, 75\$	: 1256
1C	03	AB	02	E0	005A0		BBS	#2, MOUNT_OPTIONS+3, 75\$	: 1257
17	03	AB	04	E0	005A5		BBS	#4, MOUNT_OPTIONS+3, 75\$	: 1258
12	03	AB	03	E0	005AA		BBS	#3, MOUNT_OPTIONS+3, 75\$	: 1259
		0E	05	AB	E8	005AF	BLBS	MOUNT_OPTIONS+5, 75\$	: 1260
09	05	AB	05	E0	005B3		BBS	#5, MOUNT_OPTIONS+5, 75\$	: 1261
		05	03	AB	E8	005B8	BLBS	MOUNT_OPTIONS+3, 75\$	: 1264
			02	AB	95	005BC	TSTB	MOUNT_OPTIONS+2	: 1270
			08	18	005BF		BGEQ	76\$	: 1272
		50	8F	D0	005C1	75\$:	MOVL	#7503916, R0	: 1273
				04	005C8		RET		: 1275
		50	00	D0	005C9	76\$:	MOVL	CTL\$GL PHD, PHD	: 1277
		0A	52	E9	005D0		BLBC	R2, 77\$	: 1278
06		60	03	E0	005D3		BBS	#3, (PHD), 77\$	: 1280
		50	8F	3C	005D7		MOVZWL	#10268, R0	: 1282
				04	005DC		RET		: 1283
		0A	51	E9	005DD	77\$:	BLBC	R1, 78\$	: 1284
06		60	02	E0	005E0		BBS	#2, (PHD), 78\$	: 1285
		50	8F	3C	005E4		MOVZWL	#10260, R0	: 1287
				04	005E9		RET		: 1289
0A	03	AB	02	E0	005EA	78\$:	BBS	#2, MOUNT_OPTIONS+3, 79\$	: 1290
05	03	AB	03	E0	005EF		BBS	#3, MOUNT_OPTIONS+3, 79\$	: 1292
18	03	AB	04	E1	005F4		BBC	#4, MOUNT_OPTIONS+3, 81\$	: 1302
06		60	12	E0	005F9	79\$:	BBS	#18, (PHD), 80\$	: 1302
		50	8F	3C	005FD		MOVZWL	#10388, R0	: 1302
				04	00602		RET		: 1302
09	03	AB	04	E1	00603	80\$:	BBC	#4, MOUNT_OPTIONS+3, 81\$	: 1302
		06	60	E8	00608		BLBS	(PHD), 81\$	: 1302
		50	8F	3C	0060B		MOVZWL	#10244, R0	: 1302
				04	00610		RET		: 1302
		50	01	D0	00611	81\$:	MOVL	#1, R0	: 1302
				04	00614		RET		: 1302

; Routine Size: 1557 bytes, Routine Base: \$CODE\$ + 0000


```
776 1303 1 ROUTINE COPY_ITEM (ITEM, DEST_SIZE, DEST_ADDR) =
777 1304 1
778 1305 1 ++
779 1306 1 Functional description:
780 1307 1
781 1308 1     Given an item descriptor block, this routine will copy the
782 1309 1     data described by the descriptor block to an internal storage
783 1310 1     area. The item descriptor block is assumed to already have been
784 1311 1     probed for read access, and this routine is responsible for
785 1312 1     probing the actual data for read access before copying it.
786 1313 1
787 1314 1 Input:
788 1315 1
789 1316 1     ITEM          : Address of an ITEM descriptor block.
790 1317 1     DEST_SIZE     : The size, measured in bytes, of the internal storage area.
791 1318 1     DEST_ADDR     : The address of the internal storage area.
792 1319 1
793 1320 1 Implicit Input:
794 1321 1
795 1322 1     None.
796 1323 1
797 1324 1 Output:
798 1325 1
799 1326 1     None.
800 1327 1
801 1328 1 Implicit output:
802 1329 1
803 1330 1     None.
804 1331 1
805 1332 1 Side effects:
806 1333 1
807 1334 1     None.
808 1335 1
809 1336 1 Routine value:
810 1337 1
811 1338 1     $$$_ACCVIO    : The data item could not be read accessed.
812 1339 1     $$$_BADPARAM  : The data item had a zero length.
813 1340 1     $$$_BUFFEROVF : The internal storage area was not large enough
814 1341 1                   to contain the entire data item.
815 1342 1     $$$_NORMAL    : The data was successfully copied.
816 1343 1
817 1344 1 --
818 1345 2 BEGIN                                ! Start of COPY_ITEM
819 1346 2
820 1347 2 MAP
821 1348 2     ITEM          : REF BBLOCK;          ! Item descriptor block
822 1349 2
823 1350 2 BUILTIN
824 1351 2     PROBER;                                ! MACRO-32 PROBE instruction
825 1352 2
826 1353 2 LOCAL
827 1354 2     LOCAL_LENGTH,
828 1355 2     LOCAL_ADDRESS;
829 1356 2
830 1357 2
831 1358 2 ! Copy the item length and address to internal storage.
832 1359 2 ! Reject items with a length of zero or length greater than 512.
```

```
! Normal exit value
! End of COPY_ITEM
```

				007C 00000 COPY_ITEM:				
		50	04	AC	D0 00002	.WORD	Save R2,R3,R4,R5,R6	: 1303
		51	04	A0	D0 00006	MOVL	ITEM, R0	: 1361
		56		60	3C 0000A	MOVL	4(R0), LOCAL_ADDRESS	
				09	15 0000D	MOVZWL	(R0), LOCAL_LENGTH	: 1362
	00000200	8F		56	D1 0000F	BLEQ	1\$	: 1363
				04	15 00016	CMPL	LOCAL_LENGTH, #512	
		50		14	D0 00018	BLEQ	2\$	
				04	0001B	MOVL	#20, R0	: 1365
	61	56		00	0C 0001C	RET		
				04	12 00020	PROBER	#0, LOCAL_LENGTH, (LOCAL_ADDRESS)	: 1370
		50		0C	D0 00022	BNEQ	3\$	
				04	00025	MOVL	#12, R0	: 1372
08	AC	00		56	2C 00026	RET		
		61		BC	0002C	MOVCS	LOCAL_LENGTH, (LOCAL ADDRESS), #0, -	: 1379
			0C	56	D1 0002E		DEST_SIZE, @DEST_ADDR	
	08	AC		06	15 00032	CMPL	LOCAL_LENGTH, DEST_SIZE	: 1385
		50	0601	8F	3C 00034	BLEQ	4\$	
				04	00039	MOVZWL	#1537, R0	: 1387
						RET		

MOUPAR
V04-000

H 13
16-Sep-1984 01:22:31
14-Sep-1984 12:45:31

VAX-11 BLISS-32 V4.0-742
DISK\$VM\$MASTER:[MOUNT.SRC]MOUPAR.B32;1
Page 27 (5)

50

01 D0 0003A 4\$:
04 0003D
MOVL #1, R0
RET

; 1391
;

; Routine Size: 62 bytes, Routine Base: 3CODE\$ + 0615

MOU
V04

```

.PSECT _LIB$STATES,NOWRT, SHR, PIC,1

```


	00000	ACP_STB::			
			BLKB	0	
6100	00000	:TPASTYPE			
		U.5:	WORD	24832	:
00000000*	00002	:TPASADDR			:
		U.6:	LONG	<<MOUNT_OPTIONS-U.6>-4>	:
04000000	00006	:TPASMASK			:
		U.7:	LONG	67108864	:
69F8	0000A	:TPASTYPE			:
		U.8:	WORD	27128	:
0000*	0000C	:TPASSUBEXP			:
		U.10:	WORD	<<U.9-U.10>-2>	:
00000000*	0000E	:TPASADDR			:
		U.11:	LONG	<<MOUNT_OPTIONS-U.11>-4>	:
08000000	00012	:TPASMASK			:
		U.12:	LONG	134217728	:
6DF8	00016	:TPASTYPE			:
		U.13:	WORD	28152	:
0000*	00018	:TPASSUBEXP			:
		U.15:	WORD	<<U.14-U.15>-2>	:
00000000*	0001A	:TPASADDR			:
		U.16:	LONG	<<MOUNT_OPTIONS-U.16>-4>	:
10000000	0001E	:TPASMASK			:
		U.17:	LONG	268435456	:
15F7	00022	:TPASTYPE			:
		U.19:	WORD	5623	:
FFFF	00024	:TPASTARGET			:
		U.20:	WORD	-1	:
	00026	:FILENAME			:
		U.14:	BLKB	0	:
11F1	00026	:TPASTYPE			:
		U.21:	WORD	4593	:
0000*	00028	:TPASTARGET			:
		U.22:	WORD	<<U.14-U.22>-2>	:
102E	0002A	:TPASTYPE			:
		U.23:	WORD	4142	:
0000*	0002C	:TPASTARGET			:
		U.24:	WORD	<<U.14-U.24>-2>	:
103B	0002E	:TPASTYPE			:
		U.25:	WORD	4155	:
0000*	00030	:TPASTARGET			:
		U.26:	WORD	<<U.14-U.26>-2>	:
15F6	00032	:TPASTYPE			:
		U.27:	WORD	5622	:
FFFF	00034	:TPASTARGET			:
		U.28:	WORD	-1	:
	00036	:DEVICENAME			:
		U.9:	BLKB	0	:
05F1	00036	:TPASTYPE			:
		U.29:	WORD	1521	:
043A	00038	:TPASTYPE			:
		U.30:	WORD	1082	:
15F7	0003A	:TPASTYPE			:
		U.31:	WORD	5623	:
FFFF	0003C	:TPASTARGET			:
		U.32:	WORD	-1	:

.PSECT _LIB\$KEY0\$,NOWRT, SHR, PIC,1
00000 ACP_KTB::
 .BKLB 0
00000 :TPASKEY0
 U.1: .BKLB 0
0000* 00000 :TPASKEY
 U.3: .WORD <U.2-U.1>

PSECT SUMMARY

Name	Bytes	Attributes									
\$GLOBALS	444	NOVEC,	WRT,	RD	,NOEXE,NOSHR,	LCL,	REL,	CON,NOPIC,ALIGN(2)			
\$OWNS	2324	NOVEC,	WRT,	RD	,NOEXE,NOSHR,	LCL,	REL,	CON,NOPIC,ALIGN(2)			
\$CODES	1619	NOVEC,NOWRT,		RD	,EXE,NOSHR,	LCL,	REL,	CON,NOPIC,ALIGN(2)			
_LIB\$KEY0\$	2	NOVEC,NOWRT,		RD	,EXE,SHR,	LCL,	REL,	CON,PIC,ALIGN(1)			
_LIB\$STATES	62	NOVEC,NOWRT,		RD	,EXE,SHR,	LCL,	REL,	CON,PIC,ALIGN(1)			
_LIB\$KEY1\$	8	NOVEC,NOWRT,		RD	,EXE,SHR,	LCL,	REL,	CON,PIC,ALIGN(1)			

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	88	0	1000	00:02.0
-\$255\$DUA28:[SYSLIB]CLIMAC.L32;1	14	0	0	9	00:00.1
-\$255\$DUA28:[SYSLIB]TPAMAC.L32;1	42	26	61	14	00:00.2

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:MOUPAR/OBJ=OBJ\$:MOUPAR MSRC\$:MOUPAR/UPDATE=(ENH\$:MOUPAR)
Size: 1619 code + 2840 data bytes
Run Time: 00:42.0
Elapsed Time: 01:23.6
Lines/CPU Min: 2048
Lexemes/CPU-Min: 30831
Memory Used: 438 pages
Compilation Complete

0245 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

