```
EEEEEEEEEEEEEEE   RRRRRRRRRRR      FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE   RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE   RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEEEEEEEEEEE      RRRRRRRRRRRR      FFFFFFFFFFFF
EEEEEEEEEEEE      RRRRRRRRRRRR      FFFFFFFFFFFF
EEEEEEEEEEEE      RRRRRRRRRRRR      FFFFFFFFFFFF
EEE               RRR    RRR        FFF
EEE               RRR    RRR        FFF
EEE               RRR    RRR        FFF
EEE               RRR       RRR     FFF
EEE               RRR       RRR     FFF
EEE               RRR       RRR     FFF
EEEEEEEEEEEEEEE   RRR          RRR   FFF
EEEEEEEEEEEEEEE   RRR          RRR   FFF
EEEEEEEEEEEEEEE   RRR          RRR   FFF
EEEEEEEEEEEEEEE   RRR          RRR   FFF
```

```
MM      MM    OOOOOO    UU      UU  NN      NN  TTTTTTTTTT
MM      MM    OOOOOO    UU      UU  NN      NN  TTTTTTTTTT
MMMM  MMMM  OO      OO  UU      UU  NN      NN      TT
MMMM  MMMM  OO      OO  UU      UU  NN      NN      TT
MM  MM  MM  OO      OO  UU      UU  NNNN    NN      TT
MM  MM  MM  OO      OO  UU      UU  NNNN    NN      TT
MM      MM  OO      OO  UU      UU  NN  NN  NN      TT
MM      MM  OO      OO  UU      UU  NN  NN  NN      TT
MM      MM  OO      OO  UU      UU  NN    NNNN      TT
MM      MM  OO      OO  UU      UU  NN    NNNN      TT
MM      MM  OO      OO  UU      UU  NN      NN      TT      ....
MM      MM  OO      OO  UU      UU  NN      NN      TT      ....
MM      MM    OOOOOO    UUUUUUUUUU  NN      NN      TT      ....
MM      MM    OOOOOO    UUUUUUUUUU  NN      NN      TT      ....

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



```
0179
0180
0181
0182
0183
0184
0185
0186
0187
0188
0189
0190
0191
0192
0193
0194
0195
0196
0197
0198
0199
0200
0201
0202
0203
0204
0205
0206
0207
0208
0209
0210
0211
0212
0213
0214
0215
0216
0217
0218
0219
0220
0221
0222
0223
0224
0225
0226
0227
0228
0229
0230
0231
0232
0233
0234
0235
```

```
0001    C
0002    C Version:      'V04-000'
0003    C
0004    C*****************************************************************************
0005    C*                                                                           *
0006    C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0007    C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0008    C*  ALL RIGHTS RESERVED.                                                     *
0009    C*                                                                           *
0010    C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0011    C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0012    C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0013    C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0014    C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0015    C*  TRANSFERRED.                                                             *
0016    C*                                                                           *
0017    C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0018    C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0019    C*  CORPORATION.                                                             *
0020    C*                                                                           *
0021    C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0022    C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0023    C*                                                                           *
0024    C*                                                                           *
0025    C*****************************************************************************
0026    C
0027    C
0028    C       Author  BRIAN PORTER            Creation Date   1-JUN-1980
0029    C
0030    C
0031    C       Modified by:
0032    C
0033    C       V03-001 SAR0086         Sharon A. Reynolds,     20-Jun-1983
0034    C               Changed the carriage control in the 'format' statements
0035    C               for use with ERF.
0036    C
0037    C       v02-003 BP0003          Brian Porter,           16-JUN-1981
0038    C               Added '_' and ':' to device name in display.  Added
0039    C               '"'s to the label name. Added call to LOGGER.
0040    C
0041    C       v02-002 BP0002          Brian Porter,           19-MAY-1981
0042    C               Removed RETURN 1 argument from MOUNT_VOLUME and
0043    C               DISMOUNT_VOLUME.
0044    C
0045    C       v02-001 BP0001          Brian Porter,           04-FEB-1981
0046    C               Changed queuing code.
0047    C**
0048
0049
0050
0051
0052    c++
0053    c       functional description
0054    c
0055    c       This routine maintains a list of device names and associated volume
0056    c       label that is currently mounted.
0057    c
```

```
0058     c        The format of the list entries is as follows.
0059     c
0060     c        +----------------------------------------+
0061     c        I                 flink1                 I
0062     c        +----------------------------------------+
0063     c        I                 blink1                 I
0064     c        +----------------------------------------+
0065     c        I              logging sid               I
0066     c        +----------------------------------------+
0067     c        I           root name flink              I
0068     c        +----------------------------------------+
0069     c        I           root name blink              I
0070     c        +----------------------------------------+
0071     c        I           name entry count             I
0072     c        +----------------------------------------+
0073     c
0074     c
0075     c
0076     c        +----------------------------------------+
0077     c        I                 flink2                 I
0078     c        +----------------------------------------+
0079     c        I                 blink2                 I
0080     c        +----------------------------------------+
0081     c        I                                        I
0082     c        +--                                    --+
0083     c        I         16 bytes for name              I
0084     c        +--                                    --+
0085     c        I                 string                 I
0086     c        +--                                    --+
0087     c        I                                        I
0088     c        +----------------------------------------+
0089     c        I           root unit flink              I
0090     c        +----------------------------------------+
0091     c        I           root unit blink              I
0092     c        +----------------------------------------+
0093     c        I           unit entry count             I
0094     c        +----------------------------------------+
0095     c
0096     c
0097     c
0098     c        +----------------------------------------+
0099     c        I                 flink3                 I
0100     c        +----------------------------------------+
0101     c        I                 blink3                 I
0102     c        +----------------------------------------+
0103     c        I           ucb unit number              I
0104     c        +----------------------------------------+
0105     c        I                                        I
0106     c        +--                                    --+
0107     c        I        12 byte label field             I
0108     c        +--                                    --+
0109     c        I                                        I
0110     c        +----------------------------------------+
0111     c        I     ucb operation count at mount       I
0112     c        +----------------------------------------+
0113     c        I       ucb error count at mount         I
0114     c        +----------------------------------------+
```

```
0115    c
0116    c       As mount entries are encountered an appropriate list entry is created.
0117    c       When a dismount entry is encountered the entry is removed.
0118    c       If when a mount is encountered and an entry already exists then that
0119    c       entry is updated with the current ucb data.
0120    c--
0121
0122
0123
0124
0125            subroutine mount_volume (entrance,search_sid,
0126            1 search_name_length,search_name_string,search_unit,
0127            2 vcb_label,ucb_operation_count,ucb_error_count)
0128
0129
0130
0131    c++
0132    c       Functional description:
0133    c
0134    c       This entry point is called when a MOUNT VOLUME error log
0135    c       entry type is encountered.  The volume list is searched for
0136    c       an entry corresponding to SEARCH_SID,SEARCH_NAME_STRING and
0137    c       SEARCH_UNIT.  If found then the contents of the entry are updated
0138    c       with VCB_LABEL,UCB_OPERATION_COUNT and UCB_ERROR_COUNT.  It
0139    c       is assumed that the corresponding DISMOUNT VOLUME error log entry
0140    c       is not part of the current file being processed.  If an entry is not
0141    c       found then one is made and the above mentioned information stored.
0142    c       If the call for virtual memory fails then the routine exits.
0143    c--
0144
0145
0146
0147            entry dismount_volume (entrance,search_sid,
0148            1 search_name_length,search_name_string,search_unit,
0149            2 vcb_label,mount_operation_count,mount_error_count)
0150
0151
0152
0153
0154    c++
0155    c       Functional description:
0156    c
0157    c       This routine is called when a DISMOUNT VOLUME error log entry
0158    c       is encountered.  The volume list is searched for an entry corresponding
0159    c       to SEARCH_SID,SEARCH_NAME_STRING,SEARCH_UNIT and VCB_LABEL.  If an
0160    c       entry is found then the operation and error counts are returned in
0161    c       arguments MOUNT_OPERATION_COUNT and MOUNT_ERROR_COUNT. The
0162    c       entry in the mount list is then removed.  If an entry is not found then
0163    c       the routine exits.
0164    c--
0165
0166
0167
0168
0169            entry get_current_label (entrance,search_sid,
0170            1 search_name_length,search_name_string,search_unit,
0171            2 caller_label_buffer,*)
```

```
0172
0173
0174
0175
0176     c++
0177     c       Functional description:
0178     c
0179     c       This routine is called to retrieve the name of the currently
0180     c       mounted volume of a unit.  The mount list is searched using
0181     c       SEARCH_SID,SEARCH_NAME_STRING and SEARCH_UNIT.  If an entry
0182     c       is found the label field of the entry is written to
0183     c       CALLER_LABEL_BUFFER.  If an entry is not found then the routine
0184     c       exits via RETURN 1.
0185     c--
0186
0187
0188              integer*4       buffer0(2)
0189
0190              integer*4       buffer1(6)
0191
0192              integer*4       buffer2(9)
0193
0194              integer*4       buffer3(8)
0195
0196              integer*4       root_logging_sid_flink
0197
0198              integer*4       root_logging_sid_blink
0199
0200             equivalence     (buffer0(1),root_logging_sid_flink)
0201
0202             equivalence     (buffer0(2),root_logging_sid_blink)
0203
0204              integer*4       flink1
0205
0206              integer*4       blink1
0207
0208              integer*4       logging_sid
0209
0210              integer*4       root_name_flink
0211
0212              integer*4       root_name_blink
0213
0214              integer*4       name_entry_count
0215
0216             equivalence     (buffer1(1),flink1)
0217
0218             equivalence     (buffer1(2),blink1)
0219
0220             equivalence     (buffer1(3),logging_sid)
0221
0222             equivalence     (buffer1(4),root_name_flink)
0223
0224             equivalence     (buffer1(5),root_name_blink)
0225
0226             equivalence     (buffer1(6),name_entry_count)
0227
0228
```

```
0229            integer*4       flink2
0230
0231            integer*4       blink2
0232
0233            byte            name_array(16)
0234
0235            byte            name_length
0236
0237            character*15    name_string
0238
0239            integer*4       root_unit_flink
0240
0241            integer*4       root_unit_blink
0242
0243            integer*4       unit_entry_count
0244
0245            equivalence     (buffer2(1),flink2)
0246
0247            equivalence     (buffer2(2),blink2)
0248
0249            equivalence     (buffer2(3),name_array)
0250
0251            equivalence     (name_array(1),name_length)
0252
0253            equivalence     (name_array(2),name_string)
0254
0255            equivalence     (buffer2(7),root_unit_flink)
0256
0257            equivalence     (buffer2(8),root_unit_blink)
0258
0259            equivalence     (buffer2(9),unit_entry_count)
0260
0261            integer*4       flink3
0262
0263            integer*4       blink3
0264
0265            integer*4       ucb_unit_number
0266
0267            byte            label_array
0268
0269            character*12    label_string
0270
0271            integer*4       mount_ucb_operation_count
0272
0273            integer*4       mount_ucb_error_count
0274
0275            equivalence     (buffer3(1),flink3)
0276
0277            equivalence     (buffer3(2),blink3)
0278
0279            equivalence     (buffer3(3),ucb_unit_number)
0280
0281            equivalence     (buffer3(4),label_array)
0282
0283            equivalence     (label_array,label_string)
0284
0285            equivalence     (buffer3(7),mount_ucb_operation_count)
```

```
0286
0287            equivalence     (buffer3(8),mount_ucb_error_count)
0288
0289            integer*4       logging_sid_entry_count
0290
0291            integer*4       logging_sid_entry_address
0292
0293            integer*4       name_entry_address
0294
0295            integer*4       unit_entry_address
0296
0297            integer*4       entrance
0298
0299            integer*4       search_sid
0300
0301            byte            search_name_length
0302
0303            character*15    search_name_string
0304
0305            integer*2       search_unit
0306
0307            character*15    search_name
0308
0309            character*12    vcb_label
0310
0311            integer*4       ucb_operation_count
0312
0313            integer*4       ucb_error_count
0314
0315            integer*4       caller_label_buffer
0316
0317            logical*1       lib$get_vm
0318
0319
0320
0321            call movc5 (%val(search_name_length),%ref(search_name_string),%val(42),
0322           1 %val(15),%ref(search_name))
0323
0324            logging_sid_entry_address = root_logging_sid_flink
0325
0326            do 45,i = 1,logging_sid_entry_count
0327
0328            call movc3 (%val(24),%val(logging_sid_entry_address),buffer1)
0329
0330        5   if (search_sid .eq. logging_sid) then
0331
0332            name_entry_address = root_name_flink
0333
0334            do 35,j = 1,name_entry_count
0335
0336            call movc3 (%val(36),%val(name_entry_address),buffer2)
0337
0338       10   if (search_name .eq. name_string) then
0339
0340            unit_entry_address = root_unit_flink
0341
0342            do 25,k = 1,unit_entry_count
```

```
0343
0344            call movc3 (%val(32),%val(unit_entry_address),buffer3)
0345
0346    15      if (search_unit .eq. ucb_unit_number) then
0347
0348            goto (50,75,100) entrance
0349            endif
0350
0351            unit_entry_address = flink3
0352
0353    25      continue
0354
0355            goto (28,80,110) entrance
0356
0357            return
0358
0359    28      continue
0360
0361            call movc5 (%val(0),,%val(0),%val(32),buffer3)
0362
0363            if (lib$get_vm(((32+7)/8)*8,unit_entry_address)) then
0364
0365            call insque (%val(unit_entry_address),%val(root_unit_blink))
0366
0367            ucb_unit_number = search_unit
0368
0369            call movc3 (%val(24),ucb_unit_number,%val(unit_entry_address + 8))
0370
0371            unit_entry_count = unit_entry_count + 1
0372
0373            call movl (unit_entry_count,%val(name_entry_address + 32))
0374
0375            goto 15
0376            endif
0377
0378            return
0379            endif
0380
0381            name_entry_address = flink2
0382
0383    35      continue
0384
0385            goto (38,80,110) entrance
0386
0387            return
0388
0389    38      continue
0390
0391            call movc5 (%val(0),,%val(0),%val(36),buffer2)
0392
0393            if (lib$get_vm(((36+7)/8)*8,name_entry_address)) then
0394
0395            call insque (%val(name_entry_address),%val(root_name_blink))
0396
0397            name_length = search_name_length
0398
0399            name_string = search_name
```

```
0400
0401            root_unit_flink = name_entry_address + 24
0402
0403            root_unit_blink = root_unit_flink
0404
0405            call movc3 (%val(28),name_length,%val(name_entry_address + 8))
0406
0407            name_entry_count = name_entry_count + 1
0408
0409            call movl (name_entry_count,%val(logging_sid_entry_address + 20))
0410
0411            goto 10
0412            endif
0413
0414            return
0415            endif
0416
0417            logging_sid_entry_address = flink1
0418
0419    45      continue
0420
0421            goto (48,80,110) entrance
0422
0423            return
0424
0425    48      continue
0426
0427            call movc5 (%val(0),,%val(0),%val(24),buffer1)
0428
0429            if (logging_sid_entry_count .eq. 0) then
0430
0431            root_logging_sid_flink = %loc(root_logging_sid_flink)
0432
0433            root_logging_sid_blink = root_logging_sid_flink
0434            endif
0435
0436            if (lib$get_vm(((24+7)/8)*8,logging_sid_entry_address)) then
0437
0438            call insque (%val(logging_sid_entry_address),
0439            1 %val(root_logging_sid_blink))
0440
0441            logging_sid = search_sid
0442
0443            root_name_flink = logging_sid_entry_address + 12
0444
0445            root_name_blink = root_name_flink
0446
0447            call movc3 (%val(16),logging_sid,%val(logging_sid_entry_address + 8))
0448
0449            logging_sid_entry_count = logging_sid_entry_count + 1
0450
0451            goto 5
0452            endif
0453
0454            return
0455
0456    50      continue
```

```
0457
0458      c
0459      c       Action routine for MOUNT_VOLUME
0460      c
0461
0462              label_string = vcb_label
0463
0464              mount_ucb_operation_count = ucb_operation_count
0465
0466              mount_ucb_error_count = ucb_error_count
0467
0468              call movc3 (%val(20),%ref(label_string),%val(unit_entry_address + 12))
0469
0470              return
0471
0472      75      continue
0473
0474      c
0475      c       Action routine for DISMOUNT_VOLUME
0476      c
0477
0478              if (vcb_label .eq. label_string) then
0479
0480              mount_operation_count = mount_ucb_operation_count
0481
0482              mount_error_count = mount_ucb_error_count
0483
0484              call remque (%val(unit_entry_address),unit_entry_address)
0485
0486              call lib$free_vm (((32+7)/8)*8,unit_entry_address)
0487
0488              unit_entry_count = unit_entry_count - 1
0489
0490              call movl (unit_entry_count,%val(name_entry_address + 32))
0491
0492              if (unit_entry_count .eq. 0) then
0493
0494              call remque (%val(name_entry_address),name_entry_address)
0495
0496              call lib$free_vm (((36+7)/8)*8,name_entry_address)
0497
0498              name_entry_count = name_entry_count - 1
0499
0500              call movl (name_entry_count,%val(logging_sid_entry_address + 20))
0501
0502              if (name_entry_count .eq. 0) then
0503
0504              call remque (%val(logging_sid_entry_address),logging_sid_entry_address)
0505
0506              call lib$free_vm (((24+7)/8)*8,logging_sid_entry_address)
0507
0508              logging_sid_entry_count = logging_sid_entry_count - 1
0509              endif
0510              endif
0511
0512              return
0513              endif
```

```
0514
0515     80       return
0516
0517     100      continue
0518
0519     c
0520     c        Action routine for GET_CURRENT_LABEL
0521     c
0522
0523              call movc3 (%val(12),%ref(label_string),caller_label_buffer)
0524
0525              return
0526
0527     110      return 1
0528
0529              end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	846	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	12	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	504	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	1362	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-0000001B		DISMOUNT_VOLUME	0-00000036		GET_CURRENT_LABEL
0-00000000		MOUNT_VOLUME			

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000048	I*4	BLINK1	2-00000024	I*4	BLINK2
2-00000004	I*4	BLINK3	AP-00000018@	I*4	CALLER_LABEL_BUFFER
AP-00000004@	I*4	ENTRANCE	2-00000044	I*4	FLINK1
2-00000020	I*4	FLINK2	2-00000000	I*4	FLINK3
2-00000084	I*4	I	2-00000088	I*4	J
2-0000008C	I*4	K	2-0000000C	L*1	LABEL_ARRAY
2-0000000C	CHAR	LABEL_STRING	2-0000004C	I*4	LOGGING_SID
2-00000078	I*4	LOGGING_SID_ENTRY_ADDRESS	2-00000074	I*4	LOGGING_SID_ENTRY_COUNT
AP-00000020@	I*4	MOUNT_ERROR_COUNT	AP-0000001C@	I*4	MOUNT_OPERATION_COUNT
2-0000001C	I*4	MOUNT_UCB_ERROR_COUNT	2-00000018	I*4	MOUNT_UCB_OPERATION_COUNT
2-0000007C	I*4	NAME_ENTRY_ADDRESS	2-00000058	I*4	NAME_ENTRY_COUNT
2-00000028	L*1	NAME_LENGTH	2-00000029	CHAR	NAME_STRING
2-00000060	I*4	ROOT_LOGGING_SID_BLINK	2-0000005C	I*4	ROOT_LOGGING_SID_FLINK
2-00000054	I*4	ROOT_NAME_BLINK	2-00000050	I*4	ROOT_NAME_FLINK

```
  2-0000003C  I*4  ROOT_UNIT_BLINK             2-00000038  I*4  ROOT_UNIT_FLINK
  2-00000064  CHAR SEARCH_NAME                AP-0000000C@ L*1  SEARCH_NAME_LENGTH
 AP-00000010@ CHAR SEARCH_NAME_STRING         AP-00000008@ I*4  SEARCH_SID
 AP-00000014@ I*2  SEARCH_UNIT                AP-00000020@ I*4  UCB_ERROR_COUNT
 AP-0000001C@ I*4  UCB_OPERATION_COUNT         2-00000008  I*4  UCB_UNIT_NUMBER
  2-00000080  I*4  UNIT_ENTRY_ADDRESS          2-00000040  I*4  UNIT_ENTRY_COUNT
 AP-00000018@ CHAR VCB_LABEL
```

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-0000005C	I*4	BUFFER0	8	(2)
2-00000044	I*4	BUFFER1	24	(6)
2-00000020	I*4	BUFFER2	36	(9)
2-00000000	I*4	BUFFER3	32	(8)
2-00000028	L*1	NAME_ARRAY	16	(16)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
0-0000007F	5	0-000000B0	10	0-000000DF	15	**	25	0-0000010E	28	**	35
0-00000187	38	**	45	0-00000213	48	0-00000277	50	0-0000029D	75	0-00000335	80
0-00000338	100	0-0000034A	110								

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name	Type	Name	Type	Name
	INSQUE		LIB$FREE_VM	L*1	LIB$GET_VM		MOVC3		MOVC5		MOVL
	REMQUE										

```
0001
0002
0003
0004
0005            subroutine mount (lun,option)
0006
0007
0008            include 'src$:msghdr.for /nolist'
0067
0068            include 'src$:volmount.for /nolist'
0132
0133
0134
0135
0136    c++
0137    c       Functional description:
0138    c
0139    c       This routine is called by the SYE dispatcher when a MOUNT VOLUME
0140    c       error log entry is read.
0141    c--
0142
0143
0144
0145            byte            lun
0146
0147            character*1     option
0148
0149            integer*4       compress4
0150
0151            integer*4       lib$extzv
0152
0153            logical*1       str$trim
0154
0155            integer*4       emb$t_vm_label_length
0156
0157            integer*4       mount_operation_count
0158
0159            integer*4       mount_error_count
0160
0161            integer*4       volume_operation_count
0162
0163            integer*4       volume_error_count
0164
0165
0166
0167
0168            if (option .ne. 'C') then
0169
0170            call mount_volume (1,emb$l_hd_sid,emb$b_vm_naming,emb$t_vm_name,
0171            1 emb$w_vm_unit,emb$t_vm_label,emb$l_vm_oprcnt,emb$l_vm_errcnt)
0172            endif
0173
0174            if (
0175            1 option .eq. 'S'
0176            1 .or.
0177            1 option .eq. 'B'
0178            1 ) then
```

```
0179
0180            call header (lun)
0181
0182            call logger (lun,'MOUNT VOLUME')
0183
0184            call linchk (lun,4)
0185
0186            if (.not. str$trim (emb$t_vm_label,emb$t_vm_label,
0187           1 emb$t_vm_label_length)) then
0188
0189            emb$t_vm_label_length = 12
0190            endif
0191
0192            write(lun,10) emb$t_vm_name,emb$w_vm_unit,
0193           1 emb$t_vm_label(1:emb$t_vm_label_length)
0194     10     format(7' ',t8,'UNIT  ',a<emb$b_vm_namlng>,
0195           1 i<compress4 (lib$extzv(0,16,emb$w_vm_unit))>,':, VOLUME LABEL "',a,
0196           1 '"',/)
0197
0198
0199            write(lun,15) emb$l_vm_oprcnt,emb$l_vm_errcnt
0200     15     format(' ',t8,i<compress4 (emb$l_vm_oprcnt)>,
0201           1 '. QIO OPERATIONS THIS UNIT, ',i<compress4 (emb$l_vm_errcnt)>,
0202           1 '. ERRORS THIS UNIT')
0203            endif
0204
0205            return
0206
0207
0208
0209
0210            entry dismount (lun,option)
0211
0212
0213
0214     c++
0215     c      Functional description:
0216     c
0217     c      This routine is called when a DISMOUNT VOLUME error log entry is read
0218     c--
0219
0220
0221
0222            if (option .ne. 'C') then
0223
0224            mount_operation_count = -1
0225
0226            mount_error_count = -1
0227
0228            call dismount_volume (2,emb$l_hd_sid,emb$b_vm_namlng,emb$t_vm_name,
0229           1 emb$w_vm_unit,emb$t_vm_label,mount_operation_count,
0230           1 mount_error_count)
0231            endif
0232
0233            if (
0234           1 option .eq. 'S'
0235           1 .or.
```

```
0236            1 option .eq. 'B'
0237            1 ) then
0238
0239            call header (lun)
0240
0241            call logger (lun,'DISMOUNT VOLUME')
0242
0243            call linchk (lun,4)
0244
0245            if (.not. str$trim (emb$t_vm_label,emb$t_vm_label,
0246            1 emb$t_vm_label_length)) then
0247
0248            emb$t_vm_label_length = 12
0249            endif
0250
0251            write(lun,10) emb$t_vm_name,emb$w_vm_unit,
0252            1 emb$t_vm_label(1:emb$t_vm_label_length)
0253
0254            write(lun,15) emb$l_vm_oprcnt,emb$l_vm_errcnt
0255
0256            if (
0257            1 mount_operation_count .ne. -1
0258            1 .and.
0259            1 mount_error_count .ne. -1
0260            1 ) then
0261
0262            volume_operation_count = emb$l_vm_oprcnt - mount_operation_count
0263
0264            volume_error_count = emb$l_vm_errcnt - mount_error_count
0265
0266            if (volume_operation_count) 60,25,25
0267
0268    25      if (volume_error_count) 60,30,30
0269
0270    30      call linchk (lun,1)
0271
0272            write(lun,35) volume_operation_count,volume_error_count
0273    35      format(' ',t8,i<compress4 (volume_operation_count)>,
0274            1 '. QIO OPERATIONS THIS VOLUME, ',i<compress4 (volume_error_count)>,
0275            1 '. ERRORS THIS VOLUME')
0276            endif
0277            endif
0278
0279    60      return
0280
0281            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	659	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	240	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	260	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	1671	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-000000E1		DISMOUNT	0-00000000		MOUNT

VARIABLES

Address	Type	Name	Address	Type	Name
3-0000001E	L*1	EMB$B_VM_NAMLNG	3-00000000	I*4	EMB$L_HD_SID
3-00000014	I*4	EMB$L_VM_ERRCNT	3-00000018	I*4	EMB$L_VM_OPRCNT
3-00000010	I*4	EMB$L_VM_OWNUIC	3-00000032	CHAR	EMB$T_VM_LABEL
2-00000000	I*4	EMB$T_VM_LABEL_LENGTH	3-0000001F	CHAR	EMB$T_VM_NAME
3-00000004	I*2	EMB$W_HD_ENTRY	3-0000000E	I*2	EMB$W_HD_ERRSEQ
3-00000030	I*2	EMB$W_VM_NUMSET	3-0000001C	I*2	EMB$W_VM_UNIT
3-0000002E	I*2	EMB$W_VM_VOLNUM	AP-00000004@	L*1	LUN
2-00000008	I*4	MOUNT_ERROR_COUNT	2-00000004	I*4	MOUNT_OPERATION_COUNT
AP-00000008@	CHAR	OPTION	2-00000010	I*4	VOLUME_ERROR_COUNT
2-0000000C	I*4	VOLUME_OPERATION_COUNT			

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	L*1	EMB	512	(0:511)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-00000031	10'	1-00000064	15'	**	25	**	30	1-000000A8	35'	0-0000022E	60

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4		DISMOUNT_VOLUME		HEADER
I*4	LIB$EXTZV		LINCHK		LOGGER
	MOUNT_VOLUME	L*1	STR$TRIM		

COMMAND QUALIFIERS

```
  FORTRAN /LIS=LIS$:MOUNT/OBJ=OBJ$:MOUNT MSRC$:MOUNT

  /CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
  /DEBUG=(NOSYMBOLS,TRACEBACK)
  /STANDARD=(NOSYNTAX,NOSOURCE_FORM)
  /SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP)
  /F77  /NOG_FLOATING  /I4  /OPTIMIZE  /WARNINGS  /NOD_LINES  /NOCROSS_REFERENCE  /NOMACHINE_CODE  /CONTINUATIONS=19
```

COMPILATION STATISTICS

```
  Run Time:              6.10 seconds
  Elapsed Time:         15.33 seconds
  Page Faults:          182
  Dynamic Memory:       191 pages
```



```
018
018
018
018
018
018
019
019
019
019
019
019
019
019
019
019
020
020
020
020
020
020
020
020
020
020
020
021
021
021
021
021
```

0151 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MESSAGE.LIS

ML11.LIS

MSCP.LIS

MFTAPE.LIS

MOUNT.LIS

MEMORYS.LIS

MOUVXX.LIS

MCHK_DISP.LIS