







```
0000 1 .TITLE DISMOUNT - DISMOUNT A MOUNTED MASS STORAGE VOLUME
0000 2 .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6 *
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10 *
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17 *
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21 *
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 *
0000 25 *
0000 26 *****
0000 27
0000 28 ++
0000 29
0000 30 FACILITY:
0000 31
0000 32 MASS STORAGE DEVICE MANAGEMENT SUBROUTINES
0000 33
0000 34 ABSTRACT:
0000 35
0000 36 THIS ROUTINE DISMOUNTS THE INDICATED DEVICE.
0000 37
0000 38
0000 39 ENVIRONMENT:
0000 40
0000 41 VAX/VMS EXEC
0000 42 MODE = KERNEL
0000 43
0000 44
0000 45 AUTHOR: ANDREW C. GOLDSTEIN, CREATION DATE: 2-NOV-1977 14:10
0000 46
0000 47 MODIFIED BY:
0000 48
0000 49 V03-019 CDS0001 Christian D. Saether 28-Aug-1984
0000 50 Ignore SS$_VALNOTVALID errors when converting device
0000 51 lock.
0000 52
0000 53 V03-018 HH0049 Hai Huang 16-Aug-1984
0000 54 Call IOC$DALLOC_DMT routine to deallocate the device
0000 55 on dismount of a foreign volume.
0000 56
0000 57 V03-017 ACG0441 Andrew C. Goldstein, 13-Aug-1984 10:17
```

0000 58 : Issue both an IOS\_UNLOAD and IOS\_AVAILABLE to correctly  
0000 59 : release tape drives.  
0000 60 :  
0000 61 : V03-016 ACG0441 Andrew C. Goldstein, 8-Aug-1984 11:33  
0000 62 : Rework foreign volume dismount; locate all code in  
0000 63 : this module. General code cleanup.  
0000 64 :  
0000 65 : V03-015 TMK0001 Todd M. Katz 21-Apr-1984  
0000 66 : When deleting the logical name associated with a mounted volume,  
0000 67 : delete the logical name block by calling LNMSDELETE\_LNMB  
0000 68 : instead of LNMSDELETE. Doing so will ensure that this deletion  
0000 69 : takes place as if the system service \$DELLNM had been called  
0000 70 : to delete the logical name. In other words, not only will the  
0000 71 : target logical name be deleted, but so will all outer access  
0000 72 : mode aliases.  
0000 73 :  
0000 74 : V03-014 LMP0221 L. Mark Pilant, 30-Mar-1984 13:48  
0000 75 : Change UCB\$\_OWNVIC to ORB\$\_OWNER and UCB\$\_VPROT to  
0000 76 : ORB\$\_PROT.  
0000 77 :  
0000 78 : V03-013 ACG0371 Andrew C. Goldstein, 11-Nov-1983 9:32  
0000 79 : Set PHY\_IO in PCB privilege mask instead of PHD  
0000 80 :  
0000 81 : V03-012 LY0427 Larry Yetto 5-OCT-1983 14:51:12  
0000 82 : If the DELJNL service call to delete the RU journal fails  
0000 83 : then deassign the journal channel  
0000 84 :  
0000 85 : V01-011 TCM0005 Trudy C. Matthews 22-Sep-1983  
0000 86 : If device is to be deallocated on dismount, don't do it here.  
0000 87 : Wait until last channel deassign instead. This keeps the  
0000 88 : device allocated and the lock present until all activity  
0000 89 : has ceased from this mount.  
0000 90 :  
0000 91 : V03-010 TCM0004 Trudy C. Matthews 07-Sep-1983  
0000 92 : Fix bug that caused foreign disks not to be unloaded on  
0000 93 : dismount.  
0000 94 :  
0000 95 : V03-009 TCM0003 Trudy C. Matthews 22-Aug-1983  
0000 96 : Undo change made in TCM0001. If a device is dismounted and  
0000 97 : there are still channels assigned to it, we just want to  
0000 98 : deallocate the local UCB. The cluster-wide lock (if it  
0000 99 : exists) will be dequeued when the last channel is de-assigned.  
0000 100 :  
0000 101 : V03-008 TCM0002 Trudy C. Matthews 22-Jun-1983  
0000 102 : Decrement refcount when a disk is dismounted. MOUNT has  
0000 103 : been changed to increment the refcount while the disk  
0000 104 : is mounted.  
0000 105 :  
0000 106 : V03-007 ADE9006 Alan D. Eldridge 01-MAY-1983  
0000 107 : Restore PCB address (R4) on dismount of foreign devices.  
0000 108 :  
0000 109 : V03-006 STJ3103 Steven T. Jeffreys, 27-Apr-1983  
0000 110 : Delete RUJ on dismount.  
0000 111 :  
0000 112 : V03-005 DMW4034 DMWalp 26-May-1983  
0000 113 : Intergate new logical name structures.  
0000 114 :

```

0000 115 : V03-004 TCM0001 Trudy C. Matthews 21-Apr-1982
0000 116 : Call routine EXESDALLOC_DEV to deallocate a device. This
0000 117 : routine handles cluster device deallocation correctly.
0000 118 :
0000 119 : V03-003 PHL0101 Peter H. Lipman 20-Jun-1982
0000 120 : $QIOW now synchronizes the EFN and IOSB parameters
0000 121 : correctly. Eliminate the synchronization code here.
0000 122 :
0000 123 : V03-002 STJ0257 Steven T. Jeffreys, 12-Apr-1982
0000 124 : - Do not mung device allocation access mode.
0000 125 : - Make code AST reentrant. This includes the addition
0000 126 : of the local subroutine DO_10.
0000 127 :
0000 128 : V03-001 STJ0229 Steven T. Jeffreys, 23-Mar-1982
0000 129 : Clear the 'mount verification possible' bit in the VCB
0000 130 : so that $DISMOU will succeed even if no volume is present
0000 131 : in the drive (as in version 2).
0000 132 :
0000 133 : V02-008 ACG0248 Andrew C. Goldstein, 23-Dec-1981 11:56
0000 134 : Fix logical name interlocks
0000 135 :
0000 136 : V02-007 ACG0226 Andrew C. Goldstein, 24-Nov-1981 22:29
0000 137 : Issue IOS_AVAILABLE on DISMOUNT/NOUNLOAD
0000 138 :
0000 139 : V0006 STJ0138 Steven T. Jeffreys, 12-Nov-1981
0000 140 : Use IOC$CVT_DEVNAM to format the device name.
0000 141 :
0000 142 : V0005 ACG0062 Andrew C. Goldstein, 16-Oct-1979 13:53
0000 143 : Unload volumes mounted foreign on dismount
0000 144 :
0000 145 : V0004 ACG0003 Andrew C. Goldstein, 1-Feb-1979 11:07
0000 146 : Add handling of dummy MTL entry for volume set
0000 147 :
0000 148 : Andrew C. Goldstein, 12-Jul-78 20:08
0000 149 : V0003 - ADD ERROR LOG ENTRY FOR FOREIGN DISMOUNT
0000 150 :
0000 151 : **
0000 152 :
0000 153 :
0000 154 : Define system control blocks
0000 155 :
0000 156 : $DDBDEF ; DDB
0000 157 : $DEVDEF ; device characteristics bits
0000 158 : $DCDEF ; define device types
0000 159 : $EMBETDEF ; define error log message codes
0000 160 : $EMBVMDEF ; define error log buffer format
0000 161 : $IODEF ; define I/O function codes
0000 162 : $IPLDEF ; define IPL definitions
0000 163 : $LCKDEF ; define lock manager values
0000 164 : $LKIDEF ; define codes for $GETLKI
0000 165 : $ORBDEF ; object's rights block offsets
0000 166 : $PCBDEF ; process control block
0000 167 : $PRDEF ; processor register codes
0000 168 : $PRVDEF ; privilege bit definitions
0000 169 : $MVLDEF ; mounted volume list entry
0000 170 : $SSDEF ; system service codes
0000 171 : $UCBDEF ; UCB

```

```
0000 172          $VCBDEF                ; VCB
0000 173          ;
0000 174          ; Local storage allocated on stack (addressed off R3)
0000 175          ;
00000020 0000 176 NAME_LENGTH           = 32          ; length of device name buffer
00000000 0000 177 CHANNEL              = 0           ; channel number
00000004 0000 178 DEVICE_NAME          = 4           ; string descriptor of device name
0000000C 0000 179 NAME_STRING          = 12          ; device name string buffer
0000002C 0000 180
0000002C 0000 181 LOCAL_SIZE           = 44          ; total size of stack locals
00000000 0000 182
00000000 0000 183          .PSECT Y$DISMOUNT
```

```

0000 185 :++
0000 186 :
0000 187 : FUNCTIONAL DESCRIPTION:
0000 188 :
0000 189 :     This routine dismounts the indicated mounted volume list entry.
0000 190 :     The MTL and logical name, if it still exists, are deleted, and the
0000 191 :     volume share count is decremented. If the share count goes to
0000 192 :     zero, the volume itself is dismounted.
0000 193 :
0000 194 : CALLING SEQUENCE:
0000 195 :     JSB IOC$DISMOUNT
0000 196 :
0000 197 : INPUT PARAMETERS:
0000 198 :     R3 = LBC to unload volume
0000 199 :         LBS to not unload
0000 200 :     R4 = address of process PCB
0000 201 :     R6 = address of mounted volume list entry
0000 202 :
0000 203 : IMPLICIT INPUTS:
0000 204 :     IPL - IPL$_ASTDEL
0000 205 :
0000 206 : OUTPUT PARAMETERS:
0000 207 :     R0-R2,R6 smashed, other registers preserved
0000 208 :
0000 209 : IMPLICIT OUTPUTS:
0000 210 :     NONE
0000 211 :
0000 212 : ROUTINE VALUE:
0000 213 :     $$$_NORMAL,$$$_NOIOCHAN
0000 214 :
0000 215 : SIDE EFFECTS:
0000 216 :     Volume dismounted: logical name & MTL deallocated, VCB gone or soon
0000 217 :     to go, ACP process may become deleted
0000 218 :
0000 219 :--
0000 220 :
0000 221 :
0000 222 IOC$DISMOUNT::
0000 223     PUSHR    #^M<R3,R4,R5>          ; save registers
175 0C A6 D0 0002 224     MOVL    MTL$_UCB(R6),R5        ; get UCB address
175 10 A6 D5 0006 225     TSTL    MTL$_LOGNAME(R6)       ; test address of logical name
175 1F 13 0009 226     BEQL    10$                    ; branch if none
0000 227     DSBINT  S^#IPL$_ASTDEL
0000 228     JSB    LNMSLOCKW              ; lock the table
51 10 A6 D0 0017 229     MOVL    MTL$_LOGNAME(R6),R1    ; get address of logical name
0000 230     JSB    LNMSDELETE_LNMB      ; delete the logical name
0000 231     JSB    LNMSUNLOCK          ; and unlock the table
0000 232     ENBINT
0000 233
0000 234 10$:  MOVZBL  MTL$_STATUS(R6),-(SP) ; save MTL entry status byte
0000 235     MOVL    R6,R0                ; get MTL address in R0
0000 236     BGEQ   20$                    ; branch if process space address
0000 237     BSBW   EXE$DEAPAGED          ; deallocate to system paged pool
0000 238     BRB    30$
0000 239 20$:  MOVZWL  MTL$_SIZE(R6),R1    ; get block size
53 0000 0000 '9F DE 003C 240     MOVAL   @#CTL$GQ_ALLOCREG,R3 ; and process allocation list head
0000 241     BSBW   EXE$DEALLOCATE        ; and deallocate to process pool

```



```

0046 242
0046 243
0046 244 : Now lock the I/O database mutex and decrement the volume share count.
0046 245 : If it goes to zero, mark the UCB for dismount.
0046 246
0046 247
0046 248 30$: ASSUME MTL$V_VOLSET EQ 0
50 0F 8E E8 0046 248 30$: BLBS (SP)+740$ ; branch if MTL entry was for volume set
   FF84' 30 0049 249 BSBW SCH$IOLCKW ; lock I/O database
   34 A5 D0 004C 250 MOVL UCBSL_VCB(R5),R0 ; and VCB address
   4C A0 B7 0050 251 DECB VCB$W_MCOUNT(R0) ; decrement mount count
   OC 13 0053 252 BEQL 50$ ; branch if now idle
   FF88' 30 0055 253 BSBW SCH$IOUNLOCK ; else unlock I/O database
50 01 D0 0058 254 40$: SETIPL #0
   0077 31 005E 255 MOVL #SS$_NORMAL,R0 ; set success
00 38 A5 15 E2 0061 257 BRW 130$ ; and get out
00 05 6E E9 0066 259 50$: BBSS #DEV$V_DMT,UCBSL_DEVCHAR(R5),60$ ; set mark for dismount
00 64 A5 OC E5 0069 260 60$: BLBC (SP),70$ ; branch if volume to be unloaded
   04 8A 006E 261 70$: BBCC #UCB$V_UNLOAD,UCB$W_STS(R5),70$ ; else clear unload bit
00 53 A0 0070 262 BICB2 #<1@VCB$V_MOUNTVER>,- ; clear MV bit in the VCB
00 64 A5 09 E4 0072 263 80$: VCB$B_STATUS2(R0)
   FF86' 30 0077 264 90$: BBSC #UCB$V_MOUNTING,UCB$W_STS(R5),90$ ; clean up status bits
007A 265 BSBW SCH$IOLCKW ; unlock the I/O database
007D 266 SETIPL #0
007D 267
007D 268 : Assign a channel to the device. If it is mounted Files-11, issue a dismount
007D 269 : QIO. (If it is mounted foreign, deassigning the channel will complete the
007D 270 : cleanup).
007D 271
007D 272
50 5E 2C C2 007D 272
53 5E D0 0080 273
50 20 D0 0083 274
51 OC A3 DE 0086 275
08 A3 51 D0 008A 276
   54 D4 008E 277
04 A3 FF6D' 30 0090 278
   51 D0 0093 279
   63 D4 0097 280
   0099 281
   0099 282
2B 50 E9 00A7 283
   00AA 284
   18 E1 00AA 285
08 38 A5 00AC 286
54 30 AE D0 00AF 287
   26 10 00B3 288
   14 11 00B5 289
   00B7 290
00000438 8F DD 00B7 291 100$: PUSHL #<10$ ACPCONTROL!IOSM_DMOUNT>
   63 DD 00BD 292 PUSHL CHANNEL(R3) ; push channel number
0255'CF 02 FB 00BF 293 CALLS #2,W^DO IO ; issue the dismount QIO
54 04 A3 7E 00C4 294 MOVAQ DEVICE_NAME(R3),R4 ; get address of device name descriptor
   01B3 30 00C8 295 BSBW DELETE_RUJ ; delete the recovery unit journal (ruj)
   SE 2C C0 00CB 296 110$: $DASSGN_S_CHAN=CHANNEL(R3) ; deassign the channel
   00D5 297 120$: ADDL #LOCAL_SIZE,SP ; restore stack pointer
   00D8 298

```

DISMOUNT  
V04-000

- DISMOUNT A MOUNTED MASS STORAGE VOLUME <sup>L 3</sup> 16-SEP-1984 00:02:34 VAX/VMS Macro V04-00  
5-SEP-1984 03:41:26 [SYS.SRC]DISMOUNT.MAR;1

Page 7  
(2)

38 BA 00D8 299 130\$: POPR #^M<R3,R4,R5> ; restore registers  
05 00DA 300 RSB

ER  
VC

```

00DB 302      .SBTTL DISMOUNT FOREIGN DEVICE
00DB 303      :
00DB 304      : Foreign devices are dismantled on the spot, with no interlock checks.
00DB 305      : The reason for this is that the only event we could defer the dismantl
00DB 306      : to is last channel deassign, which is not a suitable location.
00DB 307      :
00DB 308      : Construct and send the error log message signalling dismantl.
00DB 309      :
00DB 310      :
00DB 311      :
00DB 312      : .ENABLE LSB
00DB 313      FOREIGN:      : dismantl foreign device
51  3E  9A 00DB 313      MOVZBL #EMBSK_VM_LENGTH,R1      : length of error log message
      FF1F' 30 00DE 314      BSBW ERL$ALCOCEMB      : allocate an error log buffer
41 50  E9 00E1 315      BLBC R0,10$      : branch if failure
      3C  BB 00E4 316      :
      00E4 317      PUSHR #^M<R2,R3,R4,R5>      : save address of buffer and R3, R4, R5
      00E6 318      ASSUME EMBSL_VM_ERRCNT EQ EMBSL_VM_OWNUIC+4
      00E6 319      ASSUME EMBSL_VM_OPRCNT EQ EMBSL_VM_ERRCNT+4
      00E6 320      ASSUME EMBSW_VM_UNIT EQ EMBSL_VM_OPRCNT+4
      00E6 321      ASSUME EMBSB_VM_NAMLANG EQ EMBSW_VM_UNIT+2
      00E6 322      ASSUME EMBST_VM_NAMTXT EQ EMBSB_VM_NAMLANG+1
      00E6 323      ASSUME EMB$W_VM_VOLNUM EQ EMBST_VM_NAMTXT+15
      00E6 324      ASSUME EMB$W_VM_NUMSET EQ EMB$W_VM_VOLNUM+2
      00E6 325      ASSUME EMBST_VM_LABEL EQ EMB$W_VM_NUMSET+2
      00E6 326      :
04  A2  0041 8F  B0 00E6 327      MOVW #EMBSK_VD,EMBSW_VM_ENTRY(R2) ; message code = dismantl
      52  10  CC 00EC 328      ADDL #EMBSL_VM_OWNUIC,R2      : point to entries to be filled in
      50  1C  A5  DO 00EF 329      MOVL UCBSL_ORB(R5),R0      : get ORB address
      82  82  60  DO 00F3 330      MOVL ORBSL_OWNER(R0),(R2)+      : volume owner UIC
      82  0082 C5  3C 00F6 331      MOVZWL UCBSW_ERRCNT(R5),(R2)+      : volume error count
      82  70  A5  DO 00FB 332      MOVL UCBSL_OPCNT(R5),(R2)+      : volume operation count
      82  54  A5  B0 00FF 333      MOVW UCBSW_UNIT(R5),(R2)+      : unit number
7E  34  A5  14  C1 0103 334      ADDL3 #VCBSL_VOLNAME,VCBSL_VCB(R5),-(SP) ; save address of volume label
56  14  28  A5  C1 0108 335      ADDL3 UCBSL_DDB(R5),#DDBST_NAME,R6 ; calculate device name address
      50  66  9A 010D 336      MOVZBL (R6),R0      : get length of device name
62  10  00  66  50  D6 0110 337      INCL R0      : bump to include count byte
      63  9E  0C  28 011A 340      MOVCS #12,@(SP)+,(R3)      : copy device name into message
      04  BA 011E 341      CLRL (R3)+      : zero rel vol number and volume set size
      FEDD' 30 0120 342      MOVCS #12,@(SP)+,(R3)      : copy volume label
      38  BA 0123 343      POPR #^M<R2>      : recover buffer address
      0125 344      BSBW ERL$RELEASEMB      : release error log buffer and send
      0125 345      POPR #^M<R3,R4,R5>
      0125 346      :
      0125 347      : Release the device, using an unload and/or available function,
      0125 348      : depending on whether the volume is supposed to be unloaded or not.
02  0084 C4  16  E3 0127 349      10$: CLRL R2      : assume privilege bit clear
      52  D6 012D 350      BBS #PRV$V_PHY_IO,PCBSQ_PRIV(R4),20$ ; set PHY_IO privilege and test
09  64  A5  0C  E5 012F 351      INCL R2      : bit was set - save state
      01  DD 0134 352      20$: BBCC #UCBSV_UNLOAD,UCBSW_STS(R5),30$ ; branch if no unload
      63  DD 0136 353      PUSHL #IOS_UNLOAD      : set up for unload
      0255'CF 02  FB 0138 354      PUSHL CHANNEL(R3)      : push channel number
      11  DD 013C 355      CALLS #2,W^DO IO      : issue the unload or rewind QIO
0084 C4  0255'CF 02  FB 0141 356      30$: PUSHL #IOS_AVAILABLE      : now release drive
      01  16  52  FO 0146 357      PUSHL CHANNEL(R3)      : push channel number
      0146 358      CALLS #2,W^DO IO      : issue the unload or rewind QIO
      INSV R2,#PRV$V_PHY_IO,#1,PCBSQ_PRIV(R4) ; restore privilege bit

```

```

0800 8F AA 014D 359          BICW  #UCBSM_VALID,-          ; clear software volume valid.
      64 A5      0151 360          UCBSW_STS(R5)
      0153 361
      0153 362
      0153 363          ; Now complete the dismount. If the device is cluster accessible, raise
      0153 364          ; the device lock to read the value block.
      0153 365
0088 8F BB 0153 365          PUSHR #^M<R3,R7>          ; save R3 & R7
57 34 A5 DO 0157 366          MOVL  UCBSL_VCB(R5),R7      ; get VCB address
      5E 18 C2 015B 367          SUBL  #24,SP              ; allocate lock status block on stack
      56 5E DO 015E 368          MOVL  SP,R6
04 A6 20 A5 DO 0161 369          MOVL  UCBSL_LOCKID(R5),4(R6) ; get device lock ID
      6C 13 0166 370          BEQL  50$                ; branch if none - not cluster dev
61 38 A5 17 E0 0168 371          BBS   #DEV$V_ALL,UCBSL_DEVCHAR(R5),40$ ; branch if dev allocated
      50 04 DO 016D 372          MOVL  #LCK$K_PWMODE,R0    ; otherwise raise lock to PW
      0170 373          $ENQW_S LKMODE=R0,-          ; queue for the device lock
      0170 374          LKSB=(R6),-
      0170 375          EFN=S^#EXESC_SYSEFN,-
      0170 376          FLAGS=#LCK$M_CONVERT!LCK$M_VALBLK!LCK$M_SYNCSTS!LCK$M_NOQUOTA
      61 50 E9 0189 377          BLBC  R0,LOCKERR         ; bug check if error
09F0 8F 09 66 E8 018C 378          BLBS  (R6),35$
      66 B1 018F 379          CMPW  (R6),#SS$_VALNOTVALID ; Is the error simply value block not valid?
      02 13 0194 380          BEQL  35$                ; No problem.
      55 11 0196 381          BRB   LOCKERR           ; Problem.
      0198 382
      0198 383          ; Now get the lock count on the volume lock. If it is about to go to
      0198 384          ; zero, clear the value block in the device lock.
      0198 385
      7E D4 0198 386 35$: CLRL  -(SP)          ; longword for lock count
      7E 7C 019A 387          CLRQ  -(SP)          ; item list end + retlen
008 AE 9F 019C 388          PUSHAB 8(SP)          ; address of block count
02050004 8F DD 019F 389          PUSHL  #LKIS_LCKCOUNT@16!4 ; size & item code for lock count
      51 5E DO 01A5 390          MOVL  SP,R1            ; item list address
      7E 7C 01A8 391          CLRQ  -(SP)          ; IOSB
      50 5E DO 01AA 392          MOVL  SP,R0
      01AD 393          $GETLKIW_S LKIDADR=VCBSL_VOLLKID(R7),-
      01AD 394          ITMLST=(R1),-
      01AD 395          EFN=S^#EXESC_SYSEFN,-
      01AD 396          IOSB=(R0)
      29 50 E9 01C1 397          BLBC  R0,LOCKERR         ; bug check if error
      26 6E E9 01C4 398          BLBC  (SP),LOCKERR
      5E 18 C0 01C7 399          ADDL  #24,SP          ; clean IOSB & item list off stack
      8E D7 01CA 400          DECL  (SP)+          ; check lock count against 1
      06 12 01CC 401          BNEQ  50$            ; branch if other mounts exist
      08 A6 7C 01CE 402 40$: CLRQ  8(R6)          ; last mount - clear value block
      10 A6 7C 01D1 403          CLRQ  16(R6)
      01D4 404
      01D4 405          ; Now take out the I/O database mutex again, and clean out the mount.
      01D4 406          ; Release the volume lock if there is one.
      01D4 407
      FE29' 30 01D4 408 50$: BSBW  SCH$IOLOCKW          ; take I/O database mutex
50 7C A7 DO 01D7 409          MOVL  VCB$L_VOLLKID(R7),R0 ; get volume lock ID
      14 13 01DB 410          BEQL  60$            ; branch if none
      01DD 411          $DEQ_S LKID=R0        ; release it
      04 50 E8 01EA 412          BLBS  R0,60$          ; branch if OK
      01ED 413
      01ED 414          ; To here on any errors from lock management services.
      01ED 415

```

```

01ED 416 LOCKERR:
01ED 417          BUG_CHECK XQPERR,FATAL          ; unexpected lock manager error
01F1 418          ;
01F1 419          ; Clear out the UCB.
01F1 420          ;
CA 01F1 421 60$:  BICL    #<DEVSM_DMT!DEVSM_FOR!- ; clear marked for dismount, foreign,
01F2 422          DEVSM_RCK!DEVSM_WCK!- ; read/write check,
01F2 423          DEVSM_SWL!DEVSM_MNT>,- ; software write locked, and mounted
01F2 424          UCBSL_DEVCHAR(R5) ; status bit.
38 A5 C3280000 8F 01F9 425          DECW    UCBSW_REFC(R5) ; remove mount from ref count
50 5C A5 B7 01FC 426          MOVL    R7,R0 ; get address of VCB.
50 57 D0 01FF 427          CLRL    UCBSL_VCB(R5) ; clear address of VCB.
34 A5 D4 0202 428          BSBW    EXESDEANONPAGED ; deallocate VCB.
FDFB' 30 0205 429          MOVL    UCBSL_ORB(R5),R0 ; get the ORB address
50 1C A5 D0 0209 430          ;
0209 431          ASSUME  ORBSL_OWN_PROT EQ ORBSL_SYS_PROT+4
0209 432          ASSUME  ORBSL_WOR_PROT EQ ORBSL_GRP_PROT+4
0209 433          ;
18 A0 7C 0209 434          CLRQ    ORBSL_SYS_PROT(R0) ; clear out stale protection info
20 A0 7C 020C 435          CLRQ    ORBSL_GRP_PROT(R0)
60 D4 020F 436          CLRL    ORBSL_OWNER(R0) ; clear out stale owner also
0211 437          ;
0211 438          ; Release the device lock with the updated value block.
0211 439          ;
04 A6 D5 0211 440          TSTL    4(R6) ; check if we have a lock ID
50 2E 13 0214 441          BEQL    80$ ; branch if no lock
03 38 50 01 D0 0216 442          MOVL    #LCK$K_CRMODE,R0 ; use CR mode if not allocated
A5 17 E1 0219 443          BBC     #DEVSV_ALL,UCBSL_DEVCHAR(R5),70$ ; branch if not allocated
50 05 D0 021E 444          MOVL    #LCK$K_EXMODE,R0 ; use EX mode if allocated
0221 445 70$:  $ENQW_S LKMODE=R0,- ; convert the device lock down
0221 446          LKSB=(R6),- ; writing possibly modified value block
0221 447          EFN=S^#EXESC SYSEFN,-
0221 448          FLAGS=#LCK$M_CONVERT!LCK$M_CVTSYS!LCK$M_VALBLK!LCK$M_SYNCSTS!LCK$M_NOQUOTA
023E 449          ; Sorry about the tacky format above, but the assembler won't parse
023E 450          ; macro args broken across lines.
AC 50 E9 023E 451          BLBC    R0,LOCKERR ; bug check if error
A9 66 E9 0241 452          BLBC    (R6),LOCKERR
0244 453          ;
0244 454          ; Call routine to deallocate the device when appropriate
0244 455          ;
FDB9' 30 0244 456 8C$:  BSBW    IOCSDALLOC_DMT ; complete the deallocation now
0247 457          ;
FDB6' 30 0247 458 90$:  BSBW    SCH$IOUNLOCK ; release the I/O database mutex
024A 459          SETIPL #0 ; and drop IPL
5E 18 C0 024D 460          ADDL    #24,SP ; clean the stack
0088 8F BA 0250 461          POPR    #^M<R3,R7> ; restore R3 & R7
05 0254 462          RSB
0255 463
0255 464          .DISABLE LSB

```

```

0255 466 .SBTTL DO_IO - COMMON I/O ROUTINE
0255 467 :++
0255 468 : DO_IO
0255 469 :
0255 470 : FUNCTIONAL DESCRIPTION:
0255 471 :
0255 472 : This routine is an envelope procedure for all I/O done by this
0255 473 : module. Use a system event flag for the I/O. Since $QIOW now
0255 474 : properly waits for the combination of the event flag and IOSB
0255 475 : to be set, no special synchronization is needed here.
0255 476 :
0255 477 : INPUT:
0255 478 :
0255 479 : CHAN(AP) = channel number to use for the I/O
0255 480 : FUNC(AP) = I/O function code
0255 481 :
0255 482 : OUTPUT:
0255 483 :
0255 484 : NONE.
0255 485 :
0255 486 : ROUTINE VALUE:
0255 487 :
0255 488 : R0 = some system status code
0255 489 :--
0255 490 :
0255 491 :
0255 492 : Useful symbols
0255 493 :
0255 494 :
00000004 0255 495 CHAN = 4 ; offset to channel number
00000008 0255 496 FUNC = 8 ; offset to I/O function code
0255 497 :
0255 498 :
52 7E 0004 0255 499 DO_IO: .WORD ^M<R2> ; common I/O routine
0257 500 MOVAQ -(SP),R2 ; reserve IOSB, address to R2
025A 501 $QIOW_S EFN=S^#EXE$C SYSEFN,- ; use system event flag
025A 502 CHAN=CHAN(AP),- ; use channel supplied by caller
025A 503 FUNC=FUNC(AP),- ; use function code supplied by caller
025A 504 IOSB=(R2) ; use local IOSB
03 50 E9 0277 505 BLBC R0,10$ ; branch if error
50 62 3C 027A 506 MOVZWL (R2),R0 ; set the return status in R0
04 027D 507 10$: RET ; return

```

```
027E 509 .SBTTL DELETE_RUJ - DELETE RECOVERY UNIT JOURNAL
027E 510 :++
027E 511 : DELETE_RUJ
027E 512 :
027E 513 : FUNCTIONAL DESCRIPTION:
027E 514 :
027E 515 : Delete the recovery unit journal on this volume.
027E 516 : Failure to do so will leave the journal file open
027E 517 : and the device marked for dismount. This routine
027E 518 : must be called after the dismount $QIO has been
027E 519 : sent to the ACP.
027E 520 :
027E 521 : INPUT:
027E 522 :
027E 523 : R4 = address of device name descriptor
027E 524 : R5 = device UCB address
027E 525 :
027E 526 : OUTPUT:
027E 527 :
027E 528 : NONE. (Contents of R0 and R1 are unpredictable)
027E 529 :
027E 530 : ROUTINE VALUE:
027E 531 :
027E 532 : NONE.
027E 533 :--
027E 534 :
027E 535 DELETE_RUJ: ; delete recovery unit journal
41 38 05 E0 027E 536 BBS #DEV$V SQD,- ; only disks have RUJ's
0280 537 UCBSL_DEVCHAR(R5),20$ ;
0283 538 :
0283 539 : Assign a channel to the RUJ. If the service fails,
0283 540 : exit immediately, as it means that no RUJ is active.
0283 541 :
0283 542 PUSH R2 ; save R2 and make local storage
52 5E 00 0285 543 MOV SP,R2 ; save SP
0288 544 $ASSJNL_S CHAN = (R2),- ; channel to journal
0288 545 JNL_TYP = #DTS_RUJNL,- ; journal type
0288 546 DEVNAM = (R4) ; device name descriptor
1B 50 E9 02A4 547 BLBC R0,10$ ; branch if error
02A7 548 :
02A7 549 : Delete the journal. The channel to the journal is
02A7 550 : deassigned in the process.
02A7 551 :
02A7 552 $DELJNL_S CHAN = (R2) ; delete the journal
0C 50 E8 02B3 553 BLBS R0, 10$ ; if success then deljnl
02B6 554 ; deassigned the channel for us
02B6 555 $DEASJNL_S CHAN = (R2) ; deassign the journal channel
05 BA 02C2 556 10$: POPR #^M<R0,R2> ; restore stack
05 05 02C4 557 20$: RSB ; return
02C5 558
02C5 559
02C5 560 .END
```

DISMOUNT  
Symbol table

```

SST1 = 00000000
BUGS_XQPERR ***** X 02
CHAN = 00000004
CHANNEL = 00000000
CJFSASSJNL ***** GX 02
CJFSDEASJNL ***** GX 02
CJFSDELJNL ***** GX 02
CTLSGQ_ALLOCREG ***** X 02
DDBST_NAME = 00000014
DELETE_RUJ = 0000027E R 02
DEVSM_DMT = 00200000
DEVSM_FOR = 01000000
DEVSM_MNT = 00080000
DEVSM_RCK = 40000000
DEVSM_SWL = 02000000
DEVSM_WCK = 80000000
DEVSV_ALL = 00000017
DEVSV_DMT = 00000015
DEVSV_FOR = 00000018
DEVSV_SQD = 00000005
DEVICE_NAME = 00000004
DO_IO = 00000255 R 02
DTS_RUJNL = 00000001
EMBSB_VM_NAMLANG = 0000001E
EMBSK_VD = 00000041
EMBSK_VM_LENGTH = 0000003E
EMBSL_VM_ERRCNT = 00000014
EMBSL_VM_OPRCNT = 00000018
EMBSL_VM_OWNUIC = 00000010
EMBST_VM_LABEL = 00000032
EMBST_VM_NAMTX = 0000001F
EMBSW_VM_ENTRY = 00000004
EMBSW_VM_NUMSET = 00000030
EMBSW_VM_UNIT = 0000001C
EMBSW_VM_VOLNUM = 0000002E
ERLSACLOCMB ***** X 02
ERLSRELEASEMB ***** X 02
EXESC_SYSEFN ***** X 02
EXESDEALLOCATE ***** X 02
EXESDEANONPAGED ***** X 02
EXESDEAPAGED ***** X 02
FOREIGN = 000000DB R 02
FUNC = 00000008
IOSM_DMOUNT = 00000400
IOS_XPCONTROL = 00000038
IOS_AVAILABLE = 00000011
IOS_UNLOAD = 00000001
IOCSVT_DEVNAM ***** X 02
IOCSDALOC_DMT ***** X 02
IOCSDISMOUNT = 00000000 RG 02
IPLS_ASTDEL = 00000002
LCKSR_CRMODE = 00000001
LCKSK_EXMODE = 00000005
LCKSK_PWMODE = 00000004
LCKSM_CONVERT = 00000002
LCKSM_CVTSYS = 00000040
LCKSM_NOQUOTA = 00000020

```

```

LCKSM_SYNCSTS = 00000008
LCKSM_VALBLK = 00000001
LKIS_CKCOUNT = 00000205
LNMSDELETE_LNMB ***** X 02
LNMSLOCKW ***** X 02
LNMSUNLOCK ***** X 02
LOCAL_SIZE = 0000002C
LOCKERR = 000001ED R 02
MTLSB_STATUS = 0000000B
MTLSL_LOGNAME = 00000010
MTLSL_UCB = 0000000C
MTLSV_VOLSET = 00000000
MTLSW_SIZE = 00000008
NAME_LENGTH = 00000020
NAME_STRING = 0000000C
ORBSL_GRP_PROT = 00000020
ORBSL_OWNER = 00000000
ORBSL_OWN_PROT = 0000001C
ORBSL_SYS_PROT = 00000018
ORBSL_WOR_PROT = 00000024
PCBSQ_PRIV = 00000084
PRS_IPL = 00000012
PRVSV_PHY_IO = 00000016
SCHSIOLOCKW ***** X 02
SCHSIGUNLOCK ***** X 02
SSS_NORMAL = 00000001
SSS_VALNOTVALID = 000009F0
SYSSASSIGN ***** GX 02
SYSSDASSGN ***** GX 02
SYSSDEQ ***** GX 02
SYSSENQW ***** GX 02
SYSSGETLKIW ***** GX 02
SYSSQIOW ***** GX 02
UCBSL_DDB = 00000028
UCBSL_DEVCHAR = 00000038
UCBSL_LOCKID = 00000020
UCBSL_OPCNT = 00000070
UCBSL_ORB = 0000001C
UCBSL_VCB = 00000034
UCBSM_VALID = 00000800
UCBSV_MOUNTING = 00000009
UCBSV_UNLOAD = 0000000C
UCBSW_ERPCNT = 00000082
UCBSW_REFC = 0000005C
UCBSW_STS = 00000064
UCBSW_UNIT = 00000054
VCBSB_STATUS2 = 00000053
VCBSL_VOLLKID = 0000007C
VCBST_VOLNAME = 00000014
VCBSV_MOUNTVER = 00000002
VCBSW_MCOUNT = 0000004C

```



-----  
! Psect synopsis !  
-----

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 ( 0.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
Y\$DISMOUNT	000002C5 ( 709.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

-----  
! Performance indicators !  
-----

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.09	00:00:01.35
Command processing	115	00:00:00.54	00:00:04.35
Pass 1	493	00:00:19.98	00:01:03.98
Symbol table sort	1	00:00:03.25	00:00:10.29
Pass 2	115	00:00:03.44	00:00:10.17
Symbol table output	13	00:00:00.12	00:00:00.53
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	770	00:00:27.44	00:01:30.69

The working set limit was 1650 pages.  
113983 bytes (223 pages) of virtual memory were used to buffer the intermediate code.  
There were 120 pages of symbol table space allocated to hold 2155 non-local and 26 local symbols.  
560 source lines were read in Pass 1, producing 15 object records in Pass 2.  
42 pages of virtual memory were used to define 41 macros.

-----  
! Macro library statistics !  
-----

Macro library name	Macros defined
\$_\$255\$DUA28:[SYS.OBJ]LIB.F_B;1	13
\$_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	25
TOTALS (all libraries)	38

2412 GETS were required to define 38 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:DISMOUNT/OBJ=OBJ\$:DISMOUNT MSRC\$:DISMOUNT/UPDATE=(ENH\$:DISMOUNT)+EXECML\$/LIB

FILEREAD LIS	FILERWTO LIS
EXCEPTMSG LIS	EXSUBROUT LIS
DISMOUNT LIS	EXCEPTION LIS
DEBUGDATA LIS	ERRORLOG LIS
DEVICEDAT LIS	DEVICE
...	...