



```

      AAAAAA      CCCCCCCC      PPPPPPPP      CCCCCCCC      TTTTTTTTTT      RRRRRRRR
      AAAAAA      CCCCCCCC      PPPPPPPP      CCCCCCCC      TTTTTTTTTT      RRRRRRRR
AA      AA      CC      PP      PP      CC      TT      RR      RR
AA      AA      CC      PP      PP      CC      TT      RR      RR
AA      AA      CC      PP      PP      CC      TT      RR      RR
AA      AA      CC      PP      PP      CC      TT      RR      RR
AA      AA      CC      PPPPPPPP      CC      TT      RRRRRRRR
AA      AA      CC      PPPPPPPP      CC      TT      RRRRRRRR
AAAAA      CC      PP      CC      TT      RR      RR
AAAAA      CC      PP      CC      TT      RR      RR
AA      AA      CC      PP      CC      TT      RR      RR
AA      AA      CC      PP      CC      TT      RR      RR
AA      AA      CCCCCCCC      PP      CCCCCCCC      TT      RR      RR
AA      AA      CCCCCCCC      PP      CCCCCCCC      TT      RR      RR

```

```

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

```

AC  
VC

:  
:

....  
....  
....  
....

```
1 0001 0
2 0002 0 MODULE ACPCTR (LANGUAGE (BLISS32) ,
3 0003 0 IDENT = 'V04-000'
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1 *****
8 0008 1 *
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
10 0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
11 0011 1 * ALL RIGHTS RESERVED.
12 0012 1 *
13 0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
14 0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
15 0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
16 0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
17 0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
18 0018 1 * TRANSFERRED.
19 0019 1 *
20 0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
21 0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
22 0022 1 * CORPORATION.
23 0023 1 *
24 0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
25 0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
26 0026 1 *
27 0027 1 *
28 0028 1 *****
29 0029 1
30 0030 1 **
31 0031 1
32 0032 1 FACILITY: MTAACP
33 0033 1
34 0034 1 ABSTRACT:
35 0035 1 This module handles acp control functions.
36 0036 1
37 0037 1 ENVIRONMENT:
38 0038 1
39 0039 1 Starlet operating system, including privileged system services
40 0040 1 and internal exec routines.
41 0041 1
42 0042 1 --
43 0043 1
44 0044 1
45 0045 1
46 0046 1 AUTHOR: D. H. Gillespie, CREATION DATE: 09-JUL-1977
47 0047 1
48 0048 1 MODIFIED BY:
49 0049 1
50 0050 1 V03-005 MMD0236 Meg Dumont, 4-Feb-1984 15:13
51 0051 1 Add support for FIBSC_CLSEREXCP when set with IOS_ACPCONTROL.
52 0052 1
53 0053 1 V03-004 MMD0171 Meg Dumont, 9-May-1983 15:12
54 0054 1 Fix to make USER_STATUS defined consistently within module
55 0055 1
56 0056 1 V03-003 MMD0149 Meg Dumont, 26-Apr-1983 8:51
57 0057 1 Change references to 80 to the symbol ANSI_LBLSZ
```

```

58 0058 1
59 0059 1
60 0060 1
61 0061 1
62 0062 1
63 0063 1
64 0064 1
65 0065 1
66 0066 1
67 0067 1
68 0068 1
69 0069 1
70 0070 1
71 0071 1
72 0072 1
73 0073 1
74 0074 1
75 0075 1
76 0076 1
77 0077 1
78 0078 1
79 0079 1
80 0080 1
81 0081 1
82 0082 1
83 0083 1
84 0084 1
85 0085 1
86 0086 1
87 0087 1
88 0088 1
89 0089 1
90 0090 1
91 0091 1
92 0092 1
93 0093 1
94 0094 1
95 0095 1
96 0096 1
97 0097 1
98 0098 1
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 0490 1
108 0491 1
109 0492 1
110 0493 1
111 0494 1
112 0495 1
113 0496 1
114 0497 1

V03-002 MMD0001 Meg Dumont, 3-Jan-1983 15:22
Added a call to stop access to a file if the trailer labels
have been read.

V03-001 STJ0309 Steven T. Jeffreys 1-Jun-1982
Added handler for REMOUNT control function. It's a NOP.

V02-011 DMW00077 David Michael Walp 8-Feb-1982
Stored account and user name during mount time

V02-010 DMW00034 David Michael Walp 15-Sep-1981
Fixed Cancel I/O vs Dismount race condition

V02-009 DMW00024 David Michael Walp 20-Jul-1981
Change to RET_FREE_PAGE to not contract region P0 every time
space is returned.

V02-008 DMW00009 David Michael Walp 14-Mar-1981
Changed calculation of CCB address

V02-007 DMW00001 David Michael Walp 11-Nov-1980
New BLISS compiler, FUNCTION declaration changed from
BBLOCK to BLOCK. Old compiler used to give a longword
with a declaration of 'BBLOCK [1]'.

V02-006 REFORMAT Maria del C. Nasr 30-Jun-1980

V02-005 MCN0017 Maria del C. Nasr 18-Jun-1980
Add a call to START_VIO after completing the next volume
write. This is part of the fix for multivolume processing,
in which a new volume should be requested when the EOT is
sensed in writing the header labels, and not wait for the
data to be written.

V02-004 SPR27361 Maria del C. Nasr 10-Jun-1980
Add a call to START_VIO after completing the next volume
read. This is part of the general fix which delays IO
posting until all IO is successfully completed.

A0103 MCN0007 Maria del C. Nasr 13-Nov-1979 19:35
Set single directory structured device bit (DEV$M_SDI)

**
LIBRARY 'SYS$LIBRARY:LIB.L32';
REQUIRE 'SRC$:MTADEF.B32';

FORWARD ROUTINE
MTA_ACPCTRL : NOPRES NOVALUE, : control function dispatch
MTA_MOUNT : NOPRES NOVALUE, : mount function
CANCEL_IO : COMMON_CALL NOVALUE, : cancel i/o control
DO_CANCEL : COMMON_CALL NOVALUE, : do actual cancelation of io
MOUNT : COMMON_CALL NOVALUE, : mount control function, kernel mode
STALL : COMMON_CALL NOVALUE, : stall cancel

```

```

: 115      0498 1      TERMINATE_VOL: COMMON_CALL NOVALUE;      ! terminate mount volume request
: 116      0499 1
: 117      0500 1      EXTERNAL ROUTINE
: 118      0501 1      CANCEL_OP_REPLY      : COMMON_CALL,      ! cancel reply from operator
: 119      0502 1      IO_DONE,      ! complete io
: 120      0503 1      NEXT_VOL_READ      : LSNEXT_VOL_READ NOVALUE, ! get next vol for read
: 121      0504 1      NEXT_VOL_WRITE     : LSNEXT_VOL_WRIT NOVALUE, ! get next vol for write
: 122      0505 1      READ_BLOCK      : COMMON_CALL,      ! read a tape block
: 123      0506 1      RET_FREE_PAGE     : COMMON_CALL,      ! return virtual page to free list
: 124      0507 1      RETURN_ACL_ERR    : COMMON_CALL,      ! return blocked virtual io in error
: 125      0508 1      SEND_ERRLOG,
: 126      0509 1      SPACE_TM      : COMMON_CALL,      ! space tape mark
: 127      0510 1      START_VIO      : COMMON_CALL,      ! start up virtual io
: 128      0511 1      STOP_VIO      : COMMON_CALL,      ! Disallow VIO's
: 129      0512 1      SYSSBIOW     : ADDRESSING MODE (ABSOLUTE),
: 130      0513 1      ZERO_CHANNEL    : COMMON_CALL;      ! zero channel
: 131      0514 1
: 132      0515 1      EXTERNAL
: 133      0516 1      SCH$GL PCBVEC     : REF VECTOR ADDRESSING_MODE (ABSOLUTE),
: 134      0517 1      CURRENT_UCB    : REF BBLOCK,
: 135      0518 1      CURRENT_WCB    : REF BBLOCK,      ! address of current window control block
: 136      0519 1      HDR1          : REF BBLOCK,      ! hdr1(eof1) label
: 137      0520 1      IO_CHANNEL,
: 138      0521 1      IO_PACKET     : REF BBLOCK,      ! address of current io packet
: 139      0522 1      USER_STATUS   : VECTOR [2];      ! address of user status
: 140      0523 1

```

```

142 0524 1 GLOBAL ROUTINE MTA_ACPCNTRL : NOPRES NOVALUE =
143 0525 1
144 0526 1 :++
145 0527 1
146 0528 1 FUNCTIONAL DESCRIPTION:
147 0529 1 This routine handles the acp control function.
148 0530 1
149 0531 1 CALLING SEQUENCE:
150 0532 1 MTA_ACPCNTRL()
151 0533 1
152 0534 1 INPUT PARAMETERS:
153 0535 1 None
154 0536 1
155 0537 1 IMPLICIT INPUTS:
156 0538 1 CURRENT_UCB - address of current unit control block
157 0539 1 CURRENT_VCB - address of current volume control block
158 0540 1 IO_PACKET - address of current io request packet
159 0541 1 QUEUE_HEAD - address of acp queue
160 0542 1
161 0543 1 OUTPUT PARAMETERS:
162 0544 1 None
163 0545 1
164 0546 1 IMPLICIT OUTPUTS:
165 0547 1 LOCAL_FIB - copy of user's fib
166 0548 1
167 0549 1 ROUTINE VALUE:
168 0550 1 None
169 0551 1
170 0552 1 SIDE EFFECTS:
171 0553 1 None
172 0554 1
173 0555 1 --
174 0556 1
175 0557 2 BEGIN
176 0558 2
177 0559 2 EXTERNAL REGISTER
178 0560 2 COMMON_REG;
179 0561 2
180 0562 2 EXTERNAL ROUTINE
181 0563 2 ISSUE_IO : L$ISSUE_IO, ! Send an io to the tape drive
182 0564 2 GET_FIB : COMMON_CALL, ! get user's file information block
183 0565 2 POSITION_TO_END : COMMON_CALL, ! position volume set to end
184 0566 2 SPACE_IN_FILE : COMMON_CALL, ! space within file
185 0567 2 REWIND_FILE : COMMON_CALL, ! rewind file
186 0568 2 REWIND_VOL_SET : COMMON_CALL; ! rewind volume set
187 0569 2
188 0570 2 EXTERNAL
189 0571 2
190 0572 2 ! address of current unit control block
191 0573 2
192 0574 2 CURRENT_UCB : REF BBLOCK,
193 0575 2 IO_PACKET : REF BBLOCK, ! address of current io request
194 0576 2 ! packet
195 0577 2 QUEUE_HEAD : REF BBLOCK; ! address of acp queue head
196 0578 2
197 0579 2 LOCAL
198 0580 2 FIB : REF BBLOCK, ! address of copy of user's

```

```

199 0581
200 0582 FUNCTION : BLOCK [1],
201 0583
202 0584 PACKET : REF BBLOCK;
203 0585
204 0586 PACKET = .IO PACKET;
205 0587 FUNCTION = .PACKET[IRPSW_FUNC];
206 0588
207 0589 IF .FUNCTION[IOSV_DMOUNT]
208 0590 OR
209 0591 .FUNCTION[IOSV_MOUNT]
210 0592 OR
211 0593 .FUNCTION[IOSV_REMOUNT]
212 0594 THEN
213 0595 RETURN;
214 0596
215 0597 IF NOT .PACKET[IRPSV_VIRTUAL]
216 0598 THEN
217 0599 BEGIN
218 0600 KERNEL_CALL(CANCEL_IO);
219 0601
220 0602 IF (.CURRENT_VCB[VCBSV_WAIMOUVOL]
221 0603 AND
222 0604 NOT CANCEL_OP_REPLY())
223 0605 OR
224 0606 .CURRENT_VCB[VCBSV_WAIUSRLBL]
225 0607 THEN
226 0608 BEGIN
227 0609 ERROR(SS$ CANCEL);
228 0610 KERNEL_CALL(DO_CANCEL);
229 0611 END;
230 0612
231 0613 ! Stall cancel until rewind or mount vol complete so cancels are not
232 0614 ! continuously issued.
233 0615
234 0616
235 0617 IF .CURRENT_VCB[VCBSV_WAIREWIND]
236 0618 OR
237 0619 .CURRENT_VCB[VCBSV_WAIMOUVOL]
238 0620 THEN
239 0621 KERNEL_CALL(STALL);
240 0622
241 0623 RETURN;
242 0624
243 0625 END;
244 0626
245 0627 FIB = GET_FIB(.BBLOCK[.PACKET[IRPSL_SVAPTE], AIB$ DESCRPT]);
246 0628
247 0629 IF .CURRENT_VCB[VCBSV_WAIUSRLBL]
248 0630 THEN
249 0631 ERR_EXIT(SS$ WAITUSRLBL);
250 0632
251 0633 IF .CURRENT_VCB[VCBSV_MUSTCLOSE]
252 0634 THEN
253 0635 ERR_EXIT(SS$ MUSTCLOSEFL);
254 0636
255 0637 ! Allow the user to clear the serious exception from the tape drive

```

```

! file info block
! io function code and
! modifiers
! address of io request packet
! get address of io packet
! get function code and modifiers

```

```

: 256 0638 2
: 257 0639 2
: 258 0640 2
: 259 0641 2
: 260 0642 2
: 261 0643 2
: 262 0644 2
: 263 0645 2
: 264 0646 2
: 265 0647 2
: 266 0648 2
: 267 0649 2
: 268 0650 2
: 269 0651 2
: 270 0652 2
: 271 0653 2
: 272 0654 2
: 273 0655 2
: 274 0656 2
: 275 0657 2
: 276 0658 2
: 277 0659 2
: 278 0660 2
: 279 0661 2
: 280 0662 2
: 281 0663 2
: 282 0664 2
: 283 0665 2
: 284 0666 2
: 285 0667 2
: 286 0668 2
: 287 0669 2
: 288 0670 2
: 289 0671 2
: 290 0672 2
: 291 0673 2
: 292 0674 2
: 293 0675 2
: 294 0676 2
: 295 0677 2
: 296 0678 2
: 297 0679 2
: 298 0680 2
: 299 0681 2
: 300 0682 2
: 301 0683 2
: 302 0684 2
: 303 0685 3
: 304 0686 4
: 305 0687 4
: 306 0688 4
: 307 0689 4
: 308 0690 4
: 309 0691 4
: 310 0692 4
: 311 0693 4
: 312 0694 4

```

```

: by issuing a sensemode, which is effectively a NOP. This gives the
: user the capability to write blocks beyond EOT and before the
: EOY labels. It also allows the user to read blocks beyond the
: EOT and before the EOY labels. The user is never allowed to
: read the EOY labels.
:
: Please note that the case statement works on the assumption that
: the variables are within a certain range. FIBSC_CLSEREXCP does
: not fall in that range.
IF .FIB[FIBSW_CNTRLFUNC] EQL FIBSC_CLSEREXCP
  THEN
    BEGIN
      ISSUF IO(IO$_SENSEMODE, 0, 0);
      RETURN;
    END;
CASE .FIB[FIBSW_CNTRLFUNC] FROM FIBSC_REWINDVOL TO FIBSC_REWINDFIL OF
  SET
    [FIBSC_REWINDFIL] :
      REWIND_FILE();
    [FIBSC_POSEND] :
      POSITION_TO_END();
    [FIBSC_NEXTVOL] :
      BEGIN
        ! file must be accessed
        !
        IF .CURRENT_WCB EQL 0
          THEN
            ERR_EXIT(SS$_FILNOTACC);
        ! if not in data area, not appropriate time to be doing a next
        ! volume
        !
        IF .CURRENT_VCB[VCBSB_TM] NEQ 1
          THEN
            ERR_EXIT(SS$_ILLSEQOP);
        KERNEL_CALL(STOP_VIO);
        IF .CURRENT_WCB[WCB$V_READ]
          THEN
            BEGIN
              ! read case
              ! space to trailer record
              SPACE_TM(1);
            IF NOT READ_BLOCK(.HDR1, ANSI_LBLSZ)
              THEN
                ERR_EXIT(SS$_TAPEPOSLOST);
            IF .HDR1[EO1$E01LID] EQL 'EOF1'
              THEN

```



```

: 313
: 314
: 315
: 316
: 317
: 318
: 319
: 320
: 321
: 322
: 323
: 324
: 325
: 326
: 327
: 328
: 329
: 330
: 331
: 332
: 333
: 334
: 335

```

```

0695 4
0696 4
0697 4
0698 4
0699 4
0700 4
0701 4
0702 4
0703 4
0704 4
0705 4
0706 4
0707 4
0708 4
0709 4
0710 4
0711 4
0712 4
0713 4
0714 4
0715 4
0716 4
0717 1

```

```

ERR_EXIT(SS$_ENDOFFILE);
NEXT_VOL_READ();
END
ELSE
NEXT VOL WRITE();           ! write case
KERNEL_CALL(START_VIO);     ! requeue blocked io
END;

[FIB$C SPACE] :
SPACE_IN_FILE();

[FIB$C REWINDVOL] :
REWIND_VOL_SET();

[OUTRANGE] :
ERR_EXIT(SS$_ILLCNTRFUNC);

[INRANGE] :
ERR_EXIT(SS$_ILLCNTRFUNC);
TES;
END;

```

```

.TITLE ACPCTR
.IDENT \V04-000\

.EXTRN CANCEL OP REPLY
.EXTRN IO DONE, NEXT VOL READ
.EXTRN NEXT VOL WRITE, READ BLOCK
.EXTRN RET FREE PAGE, RETURN ALL_ERR
.EXTRN SEND ERRLOG, SPACE TM
.EXTRN START VIO, STOP VIO
.EXTRN SYSSQIOW, ZERO_CHANNEL
.EXTRN SCH$GL PCBVEC, CURRENT_UCB
.EXTRN CURRENT WCB, HDR1
.EXTRN IO CHANNEL, IO PACKET
.EXTRN USER STATUS, ISSUE IO
.EXTRN GET FIB, POSITION TO END
.EXTRN SPACE IN FILE, REWIND_FILE
.EXTRN REWIND VOL_SET, QUEUE_HEAD
.EXTRN SYSSCMRNL

```

```
.PSECT $CODE$,NOWRT,2
```

```

0000 00000
52 0000G CF D0 00002
50 20 A2 3C 00007
01 50 0A E1 0000B
01 50 09 E1 00010 1$:
01 50 0B E1 00015 2$:
4D 2A A2 04 00019
7E D4 0001F

```

```

.ENTRY MTA ACPCTRL, Save nothing : 0524
MOVL IO PACKET, PACKET : 0586
MOVZWL 32(PACKET), FUNCTION : 0587
BBC #10, FUNCTION, 1$ : 0589
RET :
BBC #9, FUNCTION, 2$ : 0591
RET :
BBC #11, FUNCTION, 3$ : 0593
RET :
BBS #4, 42(PACKET), 8$ : 0597
CLRL -(SP) : 0600

```

			0000V	5E DD 00021	PUSHL SP		
				CF 9F 00023	PUSHAB CANCEL IO		
08	00000000G	9F		03 FB 00027	CALLS #3, @#SYSS\$CMKRNL		0602
	OB AB			02 E1 0002E	BBC #2, 11(CURRENT_VCB), 4\$		0604
	0000G	CF		00 FB 00033	CALLS #0, CANCEL_OP_REPLY		
		05		50 E9 00038	BLBC R0, 5\$		
16	OB AB		0830	04 E1 0003B 4\$:	BBC #4, 11(CURRENT_VCB), 6\$		0606
	0000G	CF		8F B0 00040 5\$:	MOVW #2096, USER_STATUS		0609
				7E D4 00047	CLRL -(SP)		0610
			0000V	5E DD 00049	PUSHL SP		
				CF 9F 0004B	PUSHAB DO_CANCEL		
06	00000000G	9F		03 FB 0004F	CALLS #3, @#SYSS\$CMKRNL		0617
	OB AB			03 E0 00056 6\$:	BBS #3, 11(CURRENT_VCB), 7\$		0619
01	OB AB			02 E0 0005B	BBS #2, 11(CURRENT_VCB), 7\$		
				04 00060	RET		
				7E D4 00061 7\$:	CLRL -(SP)		0621
			0000V	5E DD 00063	PUSHL SP		
				CF 9F 00065	PUSHAB STALL		
			2C	00B4 31 00069	BRW 22\$		
	0000G	CF		B2 DD 0006C 8\$:	PUSHL @44(PACKET)		0627
		52		01 FB 0006F	CALLS #1, GET_FIB		
04	OB AB			50 D0 00074	MOVL R0, FIB		
			0950	04 E1 00077	BBC #4, 11(CURRENT_VCB), 9\$		0629
04	OB AB			8F BF 0007C	CHMU #2384		0631
			0948	06 E1 00080 9\$:	BBC #6, 11(CURRENT_VCB), 10\$		0633
		11		8F BF 00085	CHMU #2376		0635
			16	A2 B1 00089 10\$:	CMPW 22(FIB), #17		0648
				7B 12 0008D	BNEQ 11\$		
				7E 7C 0008F	CLRQ -(SP)		0651
				27 DD 00091	PUSHL #39		
			0000G	30 00093	BSBW ISSUE IO		
		5E		0C C0 00096	ADDL2 #12, SP		
				04 00099	RET		0650
0089	05	01	16	A2 AF 0009A 11\$:	CASEW 22(FIB), #1, #5		0655
	001B	0015		008F 0009F 12\$:	.WORD 24\$-12\$,-		
		000F		0095 000A7	14\$-12\$,-		
					15\$-12\$,-		
					23\$-12\$,-		
					25\$-12\$,-		
					13\$-12\$		
			0086	31 000AB	BRW 25\$		0711
	0000G	CF		00 FB 000AE 13\$:	CALLS #0, REWIND_FILE		0659
				04 000B3	RET		
	0000G	CF		00 FB 000B4 14\$:	CALLS #0, POSITION_TO_END		0662
				04 000B9	RET		
			0000G	CF D5 000BA 15\$:	TSTL CURRENT_WCB		0670
				04 12 000BE	BNEQ 16\$		
			00AC	8F BF 000C0	CHMU #172		0672
		01		2E AB 91 000C4 16\$:	CMPB 46(CURRENT_VCB), #1		0678
				04 13 000C8	BEQL 17\$		
			02DC	8F BF 000CA	CHMU #732		0680
				7E D4 000CE 17\$:	CLRL -(SP)		0682
				5E DD 000D0	PUSHL SP		
			0000G	CF 9F 000D2	PUSHAB STOP_VIO		
	00000000G	9F		03 FB 000D6	CALLS #3, @#SYSS\$CMKRNL		
		50	0000G	CF D0 000DD	MOVL CURRENT_WCB, R0		0684
		2F		OB A0 E9 000E2	BLBC 11(R0), -20\$		

0000G	CF	01	DD	000E6	PUSHL	#1		0687
	7E	01	FB	000E8	CALLS	#1, SPACE_TM		
		50	8F	9A 000ED	MOVZBL	#80, -(SPT)		0689
		0000G	CF	DD 000F1	PUSHL	HDR1		
0000G	CF	02	FB	000F5	CALLS	#2, READ_BLOCK		
	04	50	E8	000FA	BLBS	R0, 18\$		
		0224	8F	BF 000FD	CHMU	#548		0691
31464F45	8F	0000G	DF	D1 00101 18\$:	CMP	@HDR1, #826691397		0693
			04	12 0010A	BNEQ	19\$		
		0870	8F	BF 0010C	CHMU	#2160		0695
		0000G	30	00110 19\$:	BSBW	NEXT_VOL_READ		0697
			03	11 00113	BRB	21\$		0684
		0000G	30	00115 20\$:	BSBW	NEXT_VOL_WRITE		0700
			7E	D4 00118 21\$:	CLRL	-(SPT)		0701
			5E	DD 0011A	PUSHL	SP		
00000000G	9F	0000G	CF	9F 0011C	PUSHAB	START_VIO		
			03	FB 00120 22\$:	CALLS	#3, @SYSSCMKRN		0655
				04 00127	RET			0705
0000G	CF		00	FB 00128 23\$:	CALLS	#0, SPACE_IN_FILE		
				04 0012D	RET			
0000G	CF		00	FB 0012E 24\$:	CALLS	#0, REWIND_VOL_SET		0708
				04 00133	RET			
		00E4	8F	BF 00134 25\$:	CHMU	#228		0714
				04 00138	RET			0717

: Routine Size: 313 bytes, Routine Base: \$CODE\$ + 0000

: 336 0718 1

```
0719 1 ROUTINE MOUNT : COMMON_CALL NOVALUE =
0720 1
0721 1 ++
0722 1
0723 1 FUNCTIONAL DESCRIPTION:
0724 1 This routine gets a virtual page for the mounted volume to use
0725 1
0726 1 CALLING SEQUENCE:
0727 1 Mount(), must be called in kernel mode
0728 1
0729 1 INPUT PARAMETERS:
0730 1 None
0731 1
0732 1 IMPLICIT INPUTS:
0733 1 CURRENT_UCB - address of current unit control block
0734 1 CURRENT_VCB - address of current volume control block
0735 1
0736 1 OUTPUT PARAMETERS:
0737 1 None
0738 1
0739 1 IMPLICIT OUTPUTS:
0740 1 Virtual page for volume to use
0741 1
0742 1 ROUTINE VALUE:
0743 1 None
0744 1
0745 1 SIDE EFFECTS:
0746 1 None
0747 1
0748 1 --
0749 1
0750 2 BEGIN
0751 2
0752 2 EXTERNAL REGISTER
0753 2 COMMON_REG;
0754 2
0755 2 EXTERNAL ROUTINE
0756 2 GET_FREE_PAGE : COMMON_CALL; ! get free virtual page
0757 2
0758 2 EXTERNAL
0759 2
0760 2 ! address of current unit control block
0761 2
0762 2 CURRENT_UCB : REF BBLOCK;
0763 2
0764 2 LOCAL
0765 2 JIB : REF BBLOCK,
0766 2 PCB : REF BBLOCK,
0767 2 VPAGE : REF BBLOCK; ! address of virtual page for volume set
0768 2
0769 2 ! get virtual page for use by the volume set
0770 2
0771 2 GET FREE PAGE(1, VPAGE);
0772 2 VPAGE[VVP$B_TYPE] = VVP_TYPE;
0773 2 INSQUE(.VPAGE, CURRENT_VCB[VCB$SL_VPFL]);
0774 2 VPAGE[VVP$SL_STALLIOFL] = VPAGE[VVP$SL_STALLIOFL];
0775 2 VPAGE[VVP$SL_STALLIOBL] = VPAGE[VVP$SL_STALLIOFL];
```

```

: 395 0776 2
: 396 0777 2
: 397 0778 2
: 398 0779 2
: 399 0780 2
: 400 0781 2
: 401 0782 2
: 402 0783 2
: 403 0784 2
: 404 0785 2
: 405 0786 2
: 406 0787 1

```

```

CURRENT_UCB[UCB$$_DEVCHAR] = .CURRENT_UCB[UCB$$_DEVCHAR]
OR
(DEV$$_MNT OR DEV$$_DIR OR DEV$$_SDI);

```

```

! save the Account and User names
PCB = .SCH$$_GL PCBVEC [ (IO_PACKET[IRP$$_PID])<0, 16> ];
JIB = .PCB [ PCB$$_JIB ];
CH$$_MOVE ( VVP$$_USERNAME, JIB [JIB$$_USERNAME], VPAGE [VVP$$_USERNAME] );
CH$$_MOVE ( VVP$$_ACCOUNT, JIB [JIB$$_ACCOUNT], VPAGE [VVP$$_ACCOUNT] );
END; ! end of page

```

```

      F
      SE
0000G CF
      50
      OA A0
      51 3C
      61
      57
      50 01A4
      60
      01A8 C7
      50 0000G
      38 A0 00080018
      51 00000000G
      50 0000G
      50
      50
      50 6140
      56 0080
      01B0 C7 0C A6
      01BC C7 18 A6
      00FC 0000 MOUNT:
      04 C2 00002
      5E DD 00005
      01 DD 00007
      02 FB 00009
      6E D0 0000E
      02 90 00011
      AB 9E 00015
      60 0E 00019
      6E D0 0001C
      C7 9E 0001F
      50 D0 00024
      50 D0 00027
      CF D0 0002C
      8F C8 00031
      9F D0 00039
      CF D0 00040
      0C C0 00045
      60 3C 00048
      C0 D0 0004B
      C0 D0 0004F
      0C 28 00054
      08 28 0005B
      04 00062
      .EXTRN GET_FREE_PAGE
      .WORD Save R2,R3,R4,R5,R6,R7
      SUBL2 #4, SP
      PUSHL SP
      PUSHL #1
      CALLS #2, GET_FREE_PAGE
      MOVL VPAGE, R0
      MOVB #2, 10(R0)
      MOVAB 60(R11), R1
      INSQUE (R0), (R1)
      MOVL VPAGE, R7
      MOVAB 420(R7), R0
      MOVL R0, (R0)
      MOVL R0, 424(R7)
      MOVL CURRENT_UCB, R0
      BISL2 #524312, 56(R0)
      MOVL @#SCH$$_GL PCBVEC, R1
      MOVL IO_PACKET, R0
      ADDL2 #12, R0
      MOVZWL (R0), R0
      MOVL (R1)[R0], PCB
      MOVL 128(PCB), JIB
      MOVC3 #12, 12(JIB), 432(R7)
      MOVC3 #8, 24(JIB), 444(R7)
      RET
: 0719
: 0771
: 0772
: 0773
: 0774
: 0775
: 0777
: 0779
: 0783
: 0784
: 0785
: 0786
: 0787

```

: Routine Size: 99 bytes. Routine Base: \$CODE\$ + C139

```

: 408 0788 1 ROUTINE CANCEL_IO : COMMON_CALL NOVALUE =
: 409 0789 1
: 410 0790 1 +-
: 411 0791 1
: 412 0792 1 FUNCTIONAL DESCRIPTION:
: 413 0793 1 This routine sets the cancel io indicator if a file is accessed.
: 414 0794 1
: 415 0795 1 CALLING SEQUENCE:
: 416 0796 1 CANCEL_IO()
: 417 0797 1
: 418 0798 1 INPUT PARAMETERS:
: 419 0799 1 None
: 420 0800 1
: 421 0801 1 IMPLICIT INPUTS:
: 422 0802 1 CURRENT_VCB - address of current volume control block
: 423 0803 1
: 424 0804 1 OUTPUT PARAMETERS:
: 425 0805 1 None
: 426 0806 1
: 427 0807 1 IMPLICIT OUTPUTS:
: 428 0808 1 None
: 429 0809 1
: 430 0810 1 ROUTINE VALUE:
: 431 0811 1 None
: 432 0812 1
: 433 0813 1 SIDE EFFECTS:
: 434 0814 1 None
: 435 0815 1
: 436 0816 1 USER ERRORS:
: 437 0817 1 None
: 438 0818 1
: 439 0819 1 --
: 440 0820 1
: 441 0821 2 BEGIN
: 442 0822 2
: 443 0823 2 EXTERNAL REGISTER
: 444 0824 2 COMMON_REG;
: 445 0825 2
: 446 0826 2 IF .CURRENT_VCB[VCBSL_WCB] NEQ 0
: 447 0827 2 OR
: 448 0828 2 .CURRENT_VCB[VCBSV_WAIREWIND]
: 449 0829 2 OR
: 450 0830 2 .CURRENT_VCB[VCBSV_WAIMOUVOL]
: 451 0831 2 THEN
: 452 0832 2
: 453 0833 2 ! remember that cancel was issued
: 454 0834 2 !
: 455 0835 2 CURRENT_VCB[VCBSV_CANCELIO] = 1;
: 456 0836 2
: 457 0837 1 END;

```

0000 00000 CANCEL\_IO:  
.WORD Save nothing

; 0788

ACPCTR  
V04-000

E 11  
16-Sep-1984 02:08:09  
14-Sep-1984 12:46:31

VAX-11 Bliss-32 V4.0-742  
[MTAACP.SRC]ACPCTR.B32;1

Page 13  
(4)

			38	AB	D5	00002	TSTL	56(CURRENT_VCB)	:	0826
				0A	12	00005	BNEQ	1\$	:	
05	0B	AB		03	E0	00007	BBS	#3, 11(CURRENT_VCB), 1\$	:	0828
04	0B	AB		02	E1	0000C	BBC	#2, 11(CURRENT_VCB), 2\$	:	0830
	0B	AB		20	88	00011 1\$:	BISB2	#32, 11(CURRENT_VCB)	:	0835
				04	00015 2\$:		RET		:	0837

: Routine Size: 22 bytes, Routine Base: \$CODE\$ + 019C

: 458 0838 1

```

460 0839 1 GLOBAL ROUTINE DO_CANCEL : COMMON_CALL NOVALUE =
461 0840 1
462 0841 1 !++
463 0842 1
464 0843 1 FUNCTIONAL DESCRIPTION:
465 0844 1 This routine cancels all blocked io and acp functions.
466 0845 1
467 0846 1 CALLING SEQUENCE:
468 0847 1 DO_CANCEL(), called in kernel mode
469 0848 1
470 0849 1 INPUT PARAMETERS:
471 0850 1 None
472 0851 1
473 0852 1 IMPLICIT INPUTS:
474 0853 1 CURRENT_VCB - address of current volume control block
475 0854 1 USER_STATUS - contains error code which blocked io should be returned with
476 0855 1
477 0856 1 OUTPUT PARAMETERS:
478 0857 1 USER_STATUS - reset to normal status
479 0858 1
480 0859 1 IMPLICIT OUTPUTS:
481 0860 1 None
482 0861 1
483 0862 1 ROUTINE VALUE:
484 0863 1 None
485 0864 1
486 0865 1 SIDE EFFECTS:
487 0866 1 None
488 0867 1
489 0868 1 USER ERRORS:
490 0869 1 None
491 0870 1
492 0871 1 --
493 0872 1
494 0873 2 BEGIN
495 0874 2
496 0875 2 EXTERNAL REGISTER
497 0876 2 COMMON_REG;
498 0877 2
499 0878 2 EXTERNAL ROUTINE
500 0879 2 CHECK_DISMOUNT : COMMON_CALL; ! get free virtual page
501 0880 2
502 0881 2 MAP
503 0882 2 USER_STATUS : LONG;
504 0883 2
505 0884 2 LOCAL
506 0885 2 ABD : REF BBLOCKVECTOR [, ABD$C_LENGTH],
507 0886 2
508 0887 2 ! address containing information about blocked request
509 0888 2
510 0889 2 BLOCK_PAGE,
511 0890 2 FUNCTION : BLOCK [1],
512 0891 2 PACKET : REF BBLOCK, ! address of io blocked request
513 0892 2 WINDOW; ! address of window for this request
514 0893 2
515 0894 2 ! If the process does not have a virtual page containing the information
516 0895 2 ! describing the blocked request then have fatal error

```



```

517 0896 2
518 0897 2
519 0898 2
520 0899 2
521 0900 2
522 0901 2
523 0902 2
524 0903 2
525 0904 2
526 0905 2
527 0906 2
528 0907 2
529 0908 2
530 0909 2
531 0910 2
532 0911 2
533 0912 2
534 0913 2
535 0914 2
536 0915 2
537 0916 2
538 0917 2
539 0918 2
540 0919 2
541 0920 2
542 0921 2
543 0922 2
544 0923 2
545 0924 2
546 0925 2
547 0926 4
548 0927 4
549 0928 4
550 0929 4
551 0930 4
552 0931 4
553 0932 4
554 0933 4
555 0934 5
556 0935 5
557 0936 5
558 0937 5
559 0938 5
560 0939 5
561 0940 5
562 0941 4
563 0942 4
564 0943 4
565 0944 3
566 0945 3
567 0946 2
568 0947 2
569 0948 2
570 0949 2
571 0950 2
572 0951 2
573 0952 2

```

```

:
IF .CURRENT_VCB[VCBSL_VPFL] EQLA .CURRENT_VCB[VCBSL_VPBL]
THEN
    BUG_CHECK(NOBPVCB);

    REMQUE(.CURRENT_VCB[VCBSL_VPBL], BLOCK_PAGE);
    PACKET = (.BLOCK_PAGE + VVP$K_LENGTH + IO_PACKET - USER_STATUS);
    RET_FREE_PAGE(.BLOCK_PAGE, FALSE); ! return page(s) to virtual memory
    RETURN_ACL_ERR(); ! return all blocked physical io in error

IF .CURRENT_VCB[VCBSV_WAIMOUVOL]
OR
.CURRENT_VCB[VCBSV_WAIREWIND]
THEN
    TERMINATE_VOL(.CURRENT_VCB[VCBSL_WCB]);

! If fib descriptor present, zero count so the fib is not returned.
! complete i/o.
:

IF .PACKET NEQ 0
THEN
    BEGIN
        :
        :
        IF .PACKET[IRPSV_COMPLX]
        THEN
            BEGIN
                FUNCTION = .PACKET[IRPSW_FUNC];
                ABD = .BBLOCK[.PACKET[IRPSL_SVAPTE], AIBSL_DESCRIPTOR];

                IF .FUNCTION[IOSV_ACCESS]
                THEN
                    ZERO_CHANNEL(.PACKET)
                ELSE
                    BEGIN
                        FUNCTION = .PACKET[IRPSV_FCODE];

                        IF .FUNCTION NEQ IOS_DEACCESS
                        THEN
                            ABD[ABD$C_WINDOW, ABD$W_COUNT] = 0;

                    END;

                ABD[ABD$C_FIB, ABD$W_COUNT] = 0;
            END;

        IO_DONE(.PACKET);
    END;

! return stalled i/o with cancel
:
WHILE 1

```

```

574 0953 DO
575 0954 BEGIN
576 0955 LOCAL
577 0956 SAVE_STATUS;
578 0957
579 0958
580 0959 IF REMQUE(.BBLOCK[.CURRENT_VCB[VCBSL_VPFL], VVPSL_STALLIOFL], PACKET)
581 0960 THEN
582 0961 EXITLOOP;
583 0962
584 0963 IF .PACKET[IRPSV_COMPLX]
585 0964 THEN
586 0965 BEGIN
587 0966 FUNCTION = .PACKET[IRPSW_FUNC];
588 0967 ABD = .BBLOCK[.PACKET[IRPSL_SVAPTE], AIBSL_DESCRIPTOR];
589 0968
590 0969 IF .FUNCTION[IOSV_ACCESS]
591 0970 THEN
592 0971 ZERO_CHANNEL(.PACKET)
593 0972 ELSE
594 0973 ABD[ABDSC_WINDOW, ABD$W_COUNT] = 0;
595 0974
596 0975 ABD[ABDSC_FIB, ABD$W_COUNT] = 0;
597 0976 END;
598 0977
599 0978 ! If this is a cancel request, return is with normal status
600 0979 !
601 0980 SAVE_STATUS = .USER_STATUS;
602 0981 FUNCTION = .PACKET[IRPSV_FCODE];
603 0982
604 0983 IF .FUNCTION EQL IOS_ACPCONTROL
605 0984 AND
606 0985 NOT .PACKET[IRPSV_VIRTUAL]
607 0986 THEN
608 0987 USER_STATUS = 1;
609 0988
610 0989 IO_DONE(.PACKET);
611 0990 USER_STATUS = .SAVE_STATUS;
612 0991 END;
613 0992
614 0993 ! If no file is accessed, turn off cancel I/O bit now.
615 0994 !
616 0995
617 0996 IF .CURRENT_VCB[VCBSL_WCB] EQL 0
618 0997 THEN
619 0998 BEGIN
620 0999 CURRENT_VCB [ VCBSV_CANCELIO ] = 0;
621 1000
622 1001 ! If while the cancel I/O was pending a dismount could have been issued
623 1002 ! and refused waiting for cancel I/O to complete. Check for dismount.
624 1003 !
625 1004 CHECK_DISMOUNT ( .BBLOCK [ .CURRENT_VCB[VCBSL_RVT], RVT$UCBLST ] );
626 1005 END;
627 1006
628 1007 CURRENT_VCB[VCBSV_WAIREWIND] = 0; ! no longer waiting
629 1008 CURRENT_VCB[VCBSV_WAIUSRLBL] = 0;
630 1009 CURRENT_VCB[VCBSV_WAIMOUVOL] = 0;

```

: 631 1010 2 ERROR(SS\$ NORMAL);  
: 632 1011 1 END;

! cancel function should complete normally

.EXTRN CHECK\_DISMOUNT, BUG\$ NOBVPVCB

			007C	00000	.ENTRY	DO CANCEL, Save R2,R3,R4,R5,R6		0839
		56	CF	9E 00002	MOVAB	USER STATUS, R6		
		40	AB	D1 00007	CMP	60(CURRENT_VCB), 64(CURRENT_VCB)		0898
			04	12 0000C	BNEQ	'\$		
				FEFF 0000E	BUGW			0900
				0000* 00010	.WORD	<BUG\$ NOBVPVCB!4>		
		50	BB	0F 00012	1\$:	064(CURRENT_VCB), BLOCK_PAGE		0902
		52	CF	40 9E 00016	MOVAB	IO PACKET+12[BLOCK_PAGE], R2		0903
		51		66 9E 0001C	MOVAB	USER STATUS, R1		
		52		51 C2 0001F	SUBL2	R1, R2		
		52		62 D0 00022	MOVL	(R2), PACKET		
				7E D4 00025	CLRL	-(SP)		0904
				50 DD 00027	PUSHL	BLOCK_PAGE		
				02 FB 00029	CALLS	#2, RET FREE PAGE		
				00 FB 0002E	CALLS	#0, RETURN ACL_ERR		0905
05		08		02 E0 00033	BBS	#2, 11(CURRENT_VCB), 2\$		0907
				03 E1 00038	BBC	#3, 11(CURRENT_VCB), 3\$		0909
				AB DD 0003D	2\$:	PUSHL 56(CURRENT_VCB)		0911
				01 FB 00040	CALLS	#1, TERMINATE_VOL		
				52 D5 00045	3\$:	TSTL PACKET		0917
				32 13 00047	BEQL	7\$		
				03 E1 00049	BBC	#3, 42(PACKET), 6\$		0924
				A2 3C 0004E	MOVZWL	32(PACKET), FUNCTION		0927
				B2 D0 00052	MOVL	044(PACKET), ABD		0928
09				06 E1 00056	BBC	#6, FUNCTION, 4\$		0930
				52 DD 0005A	PUSHL	PACKET		0932
				01 FB 0005C	CALLS	#1, ZERO_CHANNEL		
				0E 11 00061	BRB	5\$		
55		20		00 EF 00063	4\$:	EXTZV #0, #6, 32(PACKET), FUNCTION		0935
				55 D1 00069	CMP	FUNCTION, #52		0937
				03 13 0006C	BEQL	5\$		
				A3 B4 0006E	CLRW	2(ABD)		0939
				0A A3 B4 00071	5\$:	CLRW 10(ABD)		0943
				52 DD 00074	6\$:	PUSHL PACKET		0946
				01 FB 00076	CALLS	#1, IO DONE		
				AB D0 0007B	7\$:	MOVL 60(CURRENT_VCB), R0		0959
				01A4 D0 0F 0007F	REMQUE	0420(R0), PACKET		
				42 1D 00084	BVS	12\$		
				03 E1 00086	BBC	#3, 42(PACKET), 10\$		0963
				A2 3C 0008B	MOVZWL	32(PACKET), FUNCTION		0966
				B2 D0 0008F	MOVL	044(PACKET), ABD		0967
09				06 E1 00093	BBC	#6, FUNCTION, 8\$		0969
				52 DD 00097	PUSHL	PACKET		0971
				01 FB 00099	CALLS	#1, ZERO_CHANNEL		
				03 11 0009E	BRB	9\$		
				02 A3 B4 000A0	8\$:	CLRW 2(ABD)		0973
				0A A3 B4 000A3	9\$:	CLRW 10(ABD)		0975
				66 D0 000A6	10\$:	MOVL USER STATUS, SAVE STATUS		0980
55		20		00 EF 000A9	EXTZV	#0, #6, 32(PACKET), FUNCTION		0981
				55 D1 000AF	CMP	FUNCTION, #56		0983

03	2A	A2	08	12	000B2	BNEQ	11\$		
		66	04	E0	000B4	BBS	#4, 42(PACKET), 11\$		0985
			01	D0	000B9	MOVL	#1, USER_STATUS		0987
	0000G	CF	52	DD	000BC	PUSHL	PACKET	11\$:	0989
		66	01	FB	000BE	CALLS	#1, IO_DONE		
			54	D0	000C3	MOVL	SAVE_STATUS, USER_STATUS		0990
			B3	11	000C6	BRB	7\$		0952
			38	AB	D5	TSTL	56(CURRENT_VCB)	12\$:	0996
			'0	12	000CB	BNEQ	13\$		
	0B	AB	20	8A	000CD	BICB2	#32, 11(CURRENT_VCB)		0999
		50	AB	D0	000D1	MOVL	32(CURRENT_VCB), R0		1004
			44	A0	DD	PUSHL	68(R0)		
	0000G	CF	01	FB	000D8	CALLS	#1, CHECK_DISMOUNT		
	0B	AB	1C	8A	000DD	BICB2	#28, 11(CURRENT_VCB)	13\$:	1009
		66	01	B0	000E1	MOVW	#1, USER_STATUS		1010
				04	000E4	RET			1011

: Routine Size: 229 bytes, Routine Base: \$CODE\$ + 01B2

: 633 1012 1

```

635 1013 1 GLOBAL ROUTINE TERMINATE_VOL (WINDOW) : COMMON_CALL NOVALUE =
636 1014 1
637 1015 1 |++
638 1016 1
639 1017 1 | FUNCTIONAL DESCRIPTION:
640 1018 1 | This routine terminates a mount request. If a file is open
641 1019 1 | then the user must close the file. The write indicator is cleared so
642 1020 1 | that eof trailers are not written on deaccess. The volume is marked
643 1021 1 | not mounted and the volume position is marked ambiguous.
644 1022 1
645 1023 1 | CALLING SEQUENCE:
646 1024 1 | TERMINATE_MOUNT(WINDOW), called in kernel mode
647 1025 1
648 1026 1 | INPUT PARAMETERS:
649 1027 1 | ARG1 - address of window for request
650 1028 1
651 1029 1 | IMPLICIT INPUTS:
652 1030 1 | None
653 1031 1
654 1032 1 | OUTPUT PARAMETERS:
655 1033 1 | None
656 1034 1
657 1035 1 | IMPLICIT OUTPUTS:
658 1036 1 | None
659 1037 1
660 1038 1 | ROUTINE VALUE:
661 1039 1 | None
662 1040 1
663 1041 1 | SIDE EFFECTS:
664 1042 1 | None
665 1043 1
666 1044 1 | USER ERRORS:
667 1045 1 | None
668 1046 1
669 1047 1 | --
670 1048 1
671 1049 2 BEGIN
672 1050 2
673 1051 2 EXTERNAL ROUTINE
674 1052 2 GET_CCB;
675 1053 2
676 1054 2 EXTERNAL REGISTER
677 1055 2 COMMON_REG;
678 1056 2
679 1057 2 MAP
680 1058 2 WINDOW : REF BBLOCK; ! address of window control block
681 1059 2
682 1060 2 LOCAL
683 1061 2 MVL_ENTRY : REF BBLOCK; ! address of MVL entry
684 1062 2
685 1063 2 IF .WINDOW NEQ 0
686 1064 2 THEN ! a file is open
687 1065 2 BEGIN
688 1066 2 CURRENT_VCB[VCBSV_NOWRITE] = 1;
689 1067 2 CURRENT_VCB[VCBSV_MUSTCLOSE] = 1; ! the file must be closed
690 1068 2 END;
691 1069 2

```

```

692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721

```

```

IF .CURRENT_VCB[VCB$V_WAIMOUVOL]
THEN
BEGIN
LOCAL
  CCB : REF BBLOCK,
  UCB : REF VECTOR;

MVL_ENTRY = .CURRENT_VCB[VCB$MVL] + MVL$K_FIXLEN +
  ((.CURRENT_VCB[VCB$B_CUR_RVN] - 1)*MVL$K_LENGTH);
MVL_ENTRY[MVL$V_MOUNTED] = 0; ! volume is not mounted
UCB = BBLOCK[.CURRENT_VCB[VCB$RVT], RVT$M_UCBLST];
UCB = .UCB[.CURRENT_VCB[VCB$W_RVN]];
CCB = GET_CCB (.IO_CHANNEL);
CCB[CCB$M_UCB] = .UCB;
SYSSQIOW(0, .IO_CHANNEL,
  IOS_REWINDOFF
  OR
  IOSM_NOWAIT
  OR
  IOSM_CLSEREXCP, 0, 0, 0, 0, 0, 0, 0, 0, 0);
SEND_ERRLOG(0, UCB);
CURRENT_VCB[VCB$B_CUR_RVN] = 0; ! no volume is current
! no file is current, ie: start at beginning
CURRENT_VCB[VCB$M_CUR_FID] = 0;
END;
END; ! end of routine TERMINATE_MOUNT

```

```

      .EXTRN GET_CCB
      .ENTRY TERMINATE_VOL, Save R2
      TSTL WINDOW
      BEQL 1$
      BISB2 #192, 11(CURRENT_VCB)
      BBC #2, 11(CURRENT_VCB), 2$
      MOVZBL 47(CURRENT_VCB), R0
      MOVAQ #52(CURRENT_VCB)[R0], MVL_ENTRY
      ADDL2 #28, MVL_ENTRY
      BICB2 #1, 7(MVL_ENTRY)
      ADDL3 #68, 32(CURRENT_VCB), UCB
      MOVZWL 14(CURRENT_VCB), R0
      MOVL (UCB)[R0], UCB
      PUSHL IO_CHANNEL
      CALLS #1, GET_CCB
      MOVL UCB, (CCB)
      CLRQ -(SP)
      CLRQ -(SP)
      CLRQ -(SP)
      CLRQ -(SP)
      CLRQ -(SP)
      CLRL -(SP)
      MOVZWL #674, -(SP)
      PUSHL IO_CHANNEL

```

```

      0004 00000
      04 AC D5 00002
      05 13 00005
      58 0B AB C0 8F 88 00007
      0B AB 02 E1 0000C 1$:
      50 2F AB 9A 00011
      50 34 BB40 7E 00015
      50 1C C0 0001A
      52 07 A0 01 8A 0001D
      20 AB 00000044 8F C1 00021
      50 0E AB 3C 0002A
      52 6240 D0 0002E
      0000G CF DD 00032
      0000G CF 01 FB 00036
      60 52 D0 0003B
      7E 7C 0003E
      7E 7C 00040
      7E 7C 00042
      7E 7C 00044
      7E D4 00046
      7E 02A2 8F 3C 00048
      0000G CF DD 0004D

```

```

: 1013
: 1063
: 1067
: 1070
: 1079
: 1078
: 1080
: 1081
: 1082
: 1083
: 1084
: 1085
: 1089
: 1085

```

ACPCTR  
V04-000

M 11  
16-Sep-1984 02:08:09  
14-Sep-1984 12:46:31

VAX-11 Bliss-32 V4.0-742  
[MTAACP.SRC]ACPCTR.B32;1

Page 21  
(6)

00000000G	9F	7E	D4	00051	CLRL	-(SP)
		0C	FB	00053	CALLS	#12, @#SYSSQIOW
		52	DD	0005A	PUSHL	UCB
0000G	CF	7E	D4	0005C	CLRL	-(SP)
		02	FB	0005E	CALLS	#2, SEND_ERRLOG
		2F	AB	94	CLRB	47(CURRENT_VCB)
		24	AB	D4	CLRL	36(CURRENT_VCB)
				04	RET	

.....  
1091  
.....  
1092  
.....  
1096  
.....  
1099

: Routine Size: 106 bytes, Routine Base: \$CODE\$ + 0297

: 722 1100 1

```

: 724      1101 1 GLOBAL ROUTINE MTA_MOUNT : NOPRES NOVALUE =
: 725      1102 1
: 726      1103 1 |++
: 727      1104 1
: 728      1105 1 | FUNCTIONAL DESCRIPTION:
: 729      1106 1 |   This routine checks the validity of the mount request and
: 730      1107 1 |   sets up a virtual page for this volume set.
: 731      1108 1
: 732      1109 1
: 733      1110 1 | CALLING SEQUENCE:
: 734      1111 1 |   MTA_MOUNT()
: 735      1112 1
: 736      1113 1 | INPUT PARAMETERS:
: 737      1114 1 |   None
: 738      1115 1
: 739      1116 1 | IMPLICIT INPUTS:
: 740      1117 1 |   CURRENT_UCB      - address of current unit control block
: 741      1118 1 |   QUEUE_HEAD      - address of queue head for ACP
: 742      1119 1
: 743      1120 1 | OUTPUT PARAMETERS:
: 744      1121 1 |   None
: 745      1122 1
: 746      1123 1 | IMPLICIT OUTPUTS:
: 747      1124 1 |   one page of virtual memory is devoted to this volume set
: 748      1125 1
: 749      1126 1 | ROUTINE VALUE:
: 750      1127 1 |   None
: 751      1128 1
: 752      1129 1 | SIDE EFFECTS:
: 753      1130 1 |   None
: 754      1131 1
: 755      1132 1 | --
: 756      1133 1
: 757      1134 2 BEGIN
: 758      1135 2
: 759      1136 2 EXTERNAL
: 760      1137 2 CURRENT_UCB      : REF BBLOCK,
: 761      1138 2 QUEUE_HEAD      : REF BBLOCK;
: 762      1139 2
: 763      1140 2 EXTERNAL REGISTER
: 764      1141 2 COMMON_REG;
: 765      1142 2
: 766      1143 2 IF NOT .BBLOCK[CURRENT_UCB[UCB$$_DEVCHAR], DEV$$_SQD]
: 767      1144 2 OR
: 768      1145 2 .QUEUE_HEAD[AQBSB_ACPTYPE] NEQ AQBSK_MTA
: 769      1146 2 THEN
: 770      1147 2 ERR_EXIT(SS$_WRONGACP);
: 771      1148 2
: 772      1149 2 KERNEL_CALL(MOUNT);
: 773      1150 1 END;

```

! end of routine MTA\_MOUNT

50 0000G CF 0000 0000

.ENTRY MTA\_MOUNT, Save nothing  
MOVL CURRENT\_UCB, R0

: 1101  
: 1143



OB	38	A0	05	E1	00007	BBC	#5, 56(R0), 1\$	:		
		50	CF	D0	0000C	MOVL	QUÉUE HEAD, R0	:	1145	
		03	15	A0	91 00011	CMPB	21(R0T, #3	:		
			04	13	00015	BEQL	2\$	:		
			031C	8F	BF 00017	1\$:	CHMU	#796	:	1147
				7E	D4 0001B	2\$:	CLRL	-(SP)	:	1149
				5E	DD 0001D		PUSHL	SP	:	
			FE15	CF	9F 0001F		PUSHAB	MOUNT	:	
	00000000G	9F	03	FB	00023		CALLS	#3, @#SYSS\$CMKRNL	:	
				04	0002A		RET		:	1150

; Routine Size: 43 bytes, Routine Base: \$CODE\$ + 0301

; 774 1151 1

```

: 776      1152 1 ROUTINE STALL : COMMON_CALL NOVALUE =
: 777      1153 1
: 778      1154 1 ++
: 779      1155 1
: 780      1156 1 FUNCTIONAL DESCRIPTION:
: 781      1157 1
: 782      1158 1     This routine puts the cancel request packet on the stalled queue.
: 783      1159 1
: 784      1160 1 CALLING SEQUENCE:
: 785      1161 1     STALL(), called in KERNEL mode
: 786      1162 1
: 787      1163 1 INPUT PARAMETERS:
: 788      1164 1     None
: 789      1165 1
: 790      1166 1 IMPLICIT INPUTS:
: 791      1167 1     None
: 792      1168 1
: 793      1169 1 OUTPUT PARAMETERS:
: 794      1170 1     None
: 795      1171 1
: 796      1172 1 IMPLICIT OUTPUTS:
: 797      1173 1     cancel request queued to stall I/O queue
: 798      1174 1
: 799      1175 1 ROUTINE VALUE:
: 800      1176 1     None
: 801      1177 1
: 802      1178 1 SIDE EFFECTS:
: 803      1179 1     None
: 804      1180 1
: 805      1181 1 --
: 806      1182 1
: 807      1183 2 BEGIN
: 808      1184 2
: 809      1185 2 EXTERNAL
: 810      1186 2     IO_PACKET      : REF BBLOCK;          ! address of current I/O packet
: 811      1187 2
: 812      1188 2 EXTERNAL REGISTER
: 813      1189 2     COMMON_REG;
: 814      1190 2
: 815      1191 2 LOCAL
: 816      1192 2     VPAGE      : REF BBLOCK;
: 817      1193 2
: 818      1194 2     VPAGE = .CURRENT VCB[VCB$L VPFL];
: 819      1195 2     INSQUE(.IO_PACKET, .VPAGE[VVP$L_STALLIOBL]);
: 820      1196 2     IO_PACKET = 0;
: 821      1197 1     END;

```

```

                                0000 0000 STALL: .WORD Save nothing      : 1152
                                3C AB D0 0002      MOVL 60(CURRENT_VCB), VPAGE : 1194
01A8 D0 0000G DF 0E 0006      INSQUE @IO_PACKET, @424(VPAGE) : 1195
                                0000G CF D4 000D      CLRL IO_PACKET          : 1196
                                04 00011      RET                    : 1197

```

: Routine Size: 18 bytes, Routine Base: \$CODE\$ + 032C

: 822 1198 1 END  
: 823 1199 1  
: 824 1200 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	830	NOVEC,NOWRT, RD, EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	Symbols		Pages Mapped	Processing Time
	Total	Loaded Percent		
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	74 0	1000	00:01.9

COMMAND QUALIFIERS

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

: Size: 830 code + 0 data bytes  
: Run Time: 00:20.6  
: Elapsed Time: 00:56.8  
: Lines/CPU Min: 3490  
: Lexemes/CPU-Min: 19902  
: Memory Used: 158 pages  
: Compilation Complete

