

LLL	I I I I I I I I I I	BBBBBBBBBBBBBB	RRRRRRRRRRRR	TTTTTTTTTTTTTTT	LLL
LLL		BBBBBBBBBBBBBB	RRRRRRRRRRRR	TTTTTTTTTTTTTTT	LLL
LLL		BBBBBBBBBBBBBB	RRRRRRRRRRRR	TTTTTTTTTTTTTTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLL		BBB        BBB	RRR        RRR	TTT	LLL
LLLLLLLLLLLL	I I I I I I I	BBBBBBBBBBBBBB	RRR        RRR	TTT	LLLLLLLLLLLL
LLLLLLLLLLLL		BBBBBBBBBBBBBB	RRR        RRR	TTT	LLLLLLLLLLLL
LLLLLLLLLLLL		BBBBBBBBBBBBBB	RRR        RRR	TTT	LLLLLLLLLLLL

\*\*FILE\*\*ID\*\*OTSCCB

B 6

000000 TTTTTTTTTT SSSSSSSS CCCCCCCC CCCCCCCC BBBBBBBB  
000000 TTTTTTTTTT SSSSSSSS CCCCCCCC CCCCCCCC BBBBBBBB  
00 00 TT SS CC CC BB BB  
00 00 TT SSSSSS CC CC BBBBBBBB  
00 00 TT SSSSSS CC CC BBBBBBBB  
00 00 TT SS CC CC BB BB  
000000 TT SSSSSSSS CCCCCCCC CCCCCCCC BBBBBBBB  
000000 TT SSSSSSSS CCCCCCCC CCCCCCCC BBBBBBBB

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

OT  
1-  
.....

:

```

1      0 MODULE OTSSCCB (
2      0           IDENT = '1-057'
3      0           ) =
4      1 BEGIN
5
6      1 ****
7      1 *
8      1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9      1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10     1 * ALL RIGHTS RESERVED.
11     1 *
12     1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13     1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14     1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15     1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16     1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17     1 * TRANSFERRED.
18     1 *
19     1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20     1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21     1 * CORPORATION.
22     1 *
23     1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24     1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25     1 *
26     1 *
27     1 ****
28     1
29     1 **+
30     1 FACILITY: language support library
31     1
32     1 ABSTRACT:
33     1
34     1 This module supports pushing and popping of the CCB, the
35     1 common control block for the I/O part of the RTL. Currently,
36     1 only BASIC uses this module, since FORTRAN does its own
37     1 manipulations.
38     1
39     1 ENVIRONMENT: User mode, AST level or not or mixed
40     1
41     1 AUTHOR: Thomas N. Hastings, CREATION DATE: 01-June-77
42     1
43     1 MODIFIED BY:
44     1
45     1 Thomas N. Hastings, 01-June-77: VERSION 01
46     1
47     1 01 - original
48     1 0-26 - Set RMS RABSV_UIF bit TNH 19-SEP-77
49     1 0-27 - Set RMS RABSV_TPT bit (truncate on sequential $PUT not at EOF TNH 24-SEP-77
50     1 0-28 - Use FOR$SIG_R0_LUB since no LUB. TNH 24-SEP-77
51     1 0-30 - Set RAB bits for read-ahead, write-behind, locate mode JMT 21-OCT-77
52     1 0-31 - Use FOR$K_abcmnoxyz as EXTERNAL LITERALS. TNH 27-Oct-77
53     1 0-32 - Made second arg optional. TNH 9-Nov-77
54     1 0-33 - Use OTSS FATINTERR. TNH 01-Dec-77
55     1 0-34 - Clear FAB after call to LIB$GET_VH. TNH 9-Dec-77
56     1 0-35 - Call FOR$SIG_FATINT. TNH 30-Dec-77
57     1 0-36 - Have CB_POP signal FATINT if LUB not active;
           Add routine CB_CND_POP to conditionally pop if LUB active.

```

: 58      0058 1 | otherwise NO-OP (OTS exit handler calls this). JMT 10-Jan-78  
59      0059 1 | 0-37 - Remove CB\_CND\_POP; I didn't really want it, anyway... JMT 11-Jan-78  
60      0060 1 | 0-37 - Global register CCB. JMT 8-Apr-78  
61      0061 1 | 0-39 - Change to STARLET library. DGP 20-Apr-78  
62      0062 1 | 0-40 - Change REQUIRE files for VAX system build. DGP 28-Apr-78  
63      0063 1 | 0-41 - Change STARLET to RTLSTARLE to avoid conflicts. DGP 1-May-78  
64      0064 1 | 0-42 - Make JSB linkage. TNH 19-May-78  
65      0065 1 | 0-46 - Use FOR\$GET\_VA with new optional 2nd arg. TNH 21-May-78  
66      0066 1 | 0-47 - Remove setting ISB to -1. TNH 30-May-78  
67      0067 1 | 0-48 - Add sanity check of data base. TNH 10-June-78  
68      0068 1 | 0-49 - Add call to FOR\$SIG\_DATCOR. TNH 10-June-78  
69      0069 1 | 0-50 - Add FOR\$CB\_GET entry for non-shared access to OTSSA\_CUR\_LUB. TNH 2-Aug-78  
70      0070 1 | 0-52 - Fix AST re-entrant timing hole. TNH 9-Aug-78  
71      0071 1 | 0-53 - Change file name to FORCB.B32, and change the names of the  
72      0072 1 | REQUIRE files similarly. JBS 14-NOV-78  
73      0073 1 | 1-001 - Update version number and copyright notice. JBS 16-JV-78  
74      0074 1 | 1-002 - Change LUB\$B\_LUN to LUB\$W\_LUN. JBS 05-DEC-78  
75      0075 1 | 1-003 - Change REQUIRE file names from FOR... to OTS... JBS 07-DEC-78  
76      0076 1 | 1-004 - Include TNH's version, which uses a bit table to provide  
77      0077 1 | AST re-entrancy. JBS 11-DEC-78  
78      0078 1 | 1-005 - Remove REQUIRE of OTSMAC; not needed. JBS 11-DEC-78  
79      0079 1 | 1-006 - Add FOR\$CB\_NEXT, which gets the next LUN for the CLOSE loop  
80      0080 1 | in FOROPEN.B32. JBS 11-DEC-78  
81      0081 1 | 1-007 - Fix coding errors in FOR\$CB\_NEXT and make OTSSAA\_LUB\_TAB  
82      0082 1 | OWN. JBS 18-DEC-78  
83      0083 1 | 1-008 - Change file and module name to OTSCB and add specialized  
84      0084 1 | BASIC entry points. This is in preparation for recursive  
85      0085 1 | I/O. JBS 29-DEC-78  
86      0086 1 | 1-009 - Add BAS\$CB\_CLEANUP. JBS 29-DEC-78  
87      0087 1 | 1-010 - Add recursive I/O for BASIC. JBS 08-JAN-1979  
88      0088 1 | 1-011 - Divide into three modules: OTSCCB, FORCB and BASCB. This  
89      0089 1 | module, OTSCCB, contains the language-independent code.  
90      0090 1 | JBS 09-JAN-1979  
91      0091 1 | 1-012 - Restore OTSSA CUR\_LUB and set I/O Active when popping.  
92      0092 1 | JBS 15-JAN-1979  
93      0093 1 | 1-013 - Fix up some complex cases of popping recursive I/O.  
94      0094 1 | JBS 15-JAN-1979  
95      0095 1 | 1-014 - Fix an error in calling LIBSTOP. JBS 16-JAN-1979  
96      0096 1 | 1-015 - Push and Pop the RMS timeout field in the RAB. JBS 16-JAN-1979  
97      0097 1 | 1-016 - Use the DEALLOC bit in the LUB to interlock deallocation of  
98      0098 1 | the LUB/ISB/RAB rather than disabling interrupts, and be  
99      0099 1 | cleverer in other places so that interrupts need never be  
100     0100 1 | disabled. JBS 23-JAN-1979  
101     0101 1 | 1-017 - Don't clear OTSSV IOINPROG if the LUN we just popped is the same  
102     0102 1 | unit we just finished using. (This is the most common case  
103     0103 1 | of recursive I/O.) JBS 24-JAN-1979  
104     0104 1 | 1-018 - But if there is no popped unit, do clear OTSSV IOINPROG. (This is  
105     0105 1 | the most common case of non-recursive I/O!). JBS 24-JAN-1979  
106     0106 1 | 1-019 - Divide into more internal subroutines in an attempt to speed  
107     0107 1 | up the common pushing and popping cases by avoiding the  
108     0108 1 | saving of unnecessary registers. JBS 25-JAN-1979  
109     0109 1 | 1-020 - Change linkage for OTSPUSH\_CCB to JSB\_CB\_PUSH and for  
110     0110 1 | OTSPOP\_CCB to JSB\_CB\_POP. JBS 25-JAN-1979  
111     0111 1 | 1-021 - Clear length of prompt-buffer when pushing. JBS 26-JAN-1979  
112     0112 1 | 1-022 - Remove OTSCLEANUP I/O, we will clean I/O using a stack  
113     0113 1 | frame instead. JBS 26-JAN-1979  
114     0114 1 | 1-023 - Change to double dollar signs since these entry points are

: 115 0115 1 | not for use by users. JBS 26-JAN-1979  
: 116 0116 1 | 1-024 - Deallocate the LUN after the LUB/ISB/RAB has been deallocated.  
: 117 0117 1 | Note that OPEN allocates it. JBS 26-JAN-1979  
: 118 0118 1 | 1-025 - Make the table storage PIC, even though it is used by INSQUE  
: 119 0119 1 | and REMQUE instructions, by initializing it at run time.  
: 120 0120 1 | This requires disabling ASIs during initialization, but it is  
: 121 0121 1 | done only once per image activation. JBS 28-JAN-1979  
: 122 0122 1 | 1-026 - Rearrange the order of some of the manipulations to make  
: 123 0123 1 | PUSH and POP really AST re-entrant. JBS 29-JAN-1979  
: 124 0124 1 | 1-027 - Make these routines AST reentrant  
: 125 0125 1 | in the face of deallocation at AST level. JBS 31-JAN-1979  
: 126 0126 1 | 1-028 - If LUB\$V\_USER\_RBUF is set, don't deallocate the record  
: 127 0127 1 | buffer, it belongs to the user! JBS 16-FEB-1979  
: 128 0128 1 | 1-029 - Clear the buddy's buddy pointer, which points to us, when  
: 129 0129 1 | deallocating. JBS 16-FEB-1979  
: 130 0130 1 | 1-030 - Print an error message if the ISB overlaps the LUB. This can  
: 131 0131 1 | happen if the LUB is extended but the ISB is not edited to  
: 132 0132 1 | reflect it. JBS 21-MAR-1979  
: 133 0133 1 | 1-031 - Initialize LUB\$Q\_BFA\_QUEUE. JBS 05-APR-1979  
: 134 0134 1 | 1-032 - Don't free the file name string unless it has been allocated  
: 135 0135 1 | in virtual memory. JBS 10-APR-1979  
: 136 0136 1 | 1-033 - Don't free the record buffer unless it has been allocated.  
: 137 0137 1 | JBS 10-APR-1979  
: 138 0138 1 | 1-034 - Free the compiled format, if allocated. SBL 27-Apr-1979  
: 139 0139 1 | 1-035 SBL1035 - Set ISBSW\_FMT\_LEN to zero on allocation. SBL 4-May-79  
: 140 0140 1 | 1-036 - Change CASE off result of REMQUEUE to match what is  
: 141 0141 1 | actually given by that function. SBL 9-May-1979  
: 142 0142 1 | 1-037 - Change require file name to OTSSCCBREQ so as not to conflict  
: 143 0143 1 | with this module at system build time. SBL 10-May-1979  
: 144 0144 1 | 1-038 - Move clearing of ISBSW\_FMT\_LEN to allocation stage. SBL 14-May-1979  
: 145 0145 1 | 1-039 - Fix bug in compiled format deallocation. SBL 15-May-1979  
: 146 0146 1 | 1-040 - Fix another one. Length must be passed as address of a word!  
: 147 0147 1 | 1-041 - We overlooked the REMQUEUE in DEALLOCATE. SBL 17-May-1979  
: 148 0148 1 | So, we have to construct a temp. SBL 17-May-1979  
: 149 0149 1 | 1-042 - Clear ISBSW\_FMT\_LEN during PUSH, so that POP won't try  
: 150 0150 1 | to deallocate the format prematurely. JBS 29-MAY-1979  
: 151 0151 1 | 1-043 - Set up LUB\$A\_BUDDY\_PTR during allocate. JBS 30-MAY-1979  
: 152 0152 1 | 1-044 - Make much of the data structure global so it can be  
: 153 0153 1 | referenced directly by FOR\$SCB. JBS 28-JUN-1979  
: 154 0154 1 | 1-045 - Do an RMS \$WAIT if there is I/O active on the unit we  
: 155 0155 1 | are starting. JBS 25-JUL-1979  
: 156 0156 1 | 1-046 - Don't make OTSSQ\_IO\_ACTIVE global. JBS 26-JUL-1979  
: 157 0157 1 | 1-047 - Save the prompt Buffer only if it really is a prompt buffer.  
: 158 0158 1 | If it is a key buffer, the key may be in read-only storage.  
: 159 0159 1 | JBS 09-AUG-1979  
: 160 0160 1 | 1-048 - Move the global parts of the data base to OTSSCCB\_DATA  
: 161 0161 1 | so this module need not be loaded if only FORTRAN programs  
: 162 0162 1 | are in the image. JBS 16-AUG-1979  
: 163 0163 1 | 1-049 - Return CCB as 0 from POP to indicate deallocation. JBS 17-AUG-1979  
: 164 0164 1 | 1-050 - Correct an error in a comment. JBS 10-SEP-1979  
: 165 0165 1 | 1-051 - When deallocating, LUB\$A\_BUF\_BEG points to the buffer; in locate  
: 166 0166 1 | mode, LUB\$A\_RBUF\_ADR may point to RMS space. JBS 13-SEP-1979  
: 167 0167 1 | 1-052 - Remove the references to ISBSW\_FMT\_LEN; now done in FORCB.  
: 168 0168 1 | JBS 18-SEP-1979  
: 169 0169 1 | 1-053 - Remove references to LUB\$Q\_BFA\_QUEUE; no longer used.  
: 170 0170 1 | JBS 18-SEP-1979  
: 171 0171 1 | 1-054 - Correct a minor typo. JBS 24-OCT-1979

: 172 0172 1 | 1-055 - Use the new UBF cell in the LUB. JBS 13-NOV-1979  
: 173 0173 1 | 1-056 - Don't initialize LUB table entries in use by FORTRAN. JBS 14-JAN-1980  
: 174 0174 1 | 1-057 - Take out clearing of RAB\$B\_PSZ (put it in BAS\$S10\_BEG)  
: 175 0175 1 | to make locality consistent. FM 4-SEP-1980  
: 176 0176 1 | --  
: 177 0177 1 |  
: 178 0178 1 !<BLF/PAGE>

```
: 180 0179 1 |  
: 181 0180 1 | SWITCHES:  
: 182 0181 1 |  
: 183 0182 1 |  
: 184 0183 1 | SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);  
: 185 0184 1 |  
: 186 0185 1 |  
: 187 0186 1 | LINKAGES:  
: 188 0187 1 |  
: 189 0188 1 |  
: 190 0189 1 REQUIRE 'RTLIN:OTSLNK'; ! Define LINKAGES  
: 191 0618 1 |  
: 192 0619 1 | TABLE OF CONTENTS:  
: 193 0620 1 |  
: 194 0621 1 |  
: 195 0622 1 |  
: 196 0623 1 FORWARD ROUTINE  
: 197 0624 1 INITIALIZE : NOVALUE.  
: 198 0625 1 PUSH_FAKE : CALL CCB.  
: 199 0626 1 PUSH_ACTIVE : CALL CCB.  
: 200 0627 1 ALLOCATE : CALL CCB.  
: 201 0628 1 OTSS$PUSH_CCB : JSB_CCB PUSH,  
: 202 0629 1 DEALLOCATE : CALL_CCB NOVALUE,  
: 203 0630 1 POP_ACTIVE : CALL_CCB NOVALUE,  
: 204 0631 1 OTSS$POP_CCB : JSB_CCB_POP NOVALUE;  
: 205 0632 1 | Set up the LUB table and the active queue  
: 206 0633 1 | Push fake record  
: 207 0634 1 | Push active LUB  
: 208 0635 1 | Allocate LUB/ISB/RAB  
: 209 0636 1 | Get the CCB, push old use of it  
: 210 0637 1 | Deallocate LUB/ISB/RAB  
: 211 0805 1 | Pop active LUN  
: 212 0806 1 | Restore old use of CCB  
: 213 0946 1 |  
: 214 0947 1 REQUIRE 'RTLML:OTESISB'; ! get length of ISB  
: 215 1042 1 REQUIRE 'RTLML:OTSLUB'; ! get length of LUB  
: 216 1043 1 REQUIRE 'RTLIN:RTLPSECT'; ! Define DECLARE_PSECTs macro  
: 217 1141 1 REQUIRE 'RTLIN:OTSCCBREQ'; ! Define interface to OTSS$PUSH_CCB  
: 218 1142 1 LIBRARY 'RTLSTARLE'; ! STARLET library for macros and symbols  
: 219 1143 1 |  
: 220 1144 1 |  
: 221 1145 1 MACROS:  
: 222 1146 1 |  
: 223 1147 1 |  
: 224 1148 1 MACRO  
M 1149 1 TEST_LUB_ISB =  
M 1150 1 |  
M 1151 1 Give an error message if the ISB and the LUB overlap. Try to make the  
M 1152 1 message explicit enough to tell the maintainer exactly what to do, since  
M 1153 1 it will print only when the RTL is being modified by someone who does not  
M 1154 1 know about the LUB-ISB dependency, and therefore may need a lot of hand-  
M 1155 1 holding.  
M 1156 1 |  
M 1157 1 |  
M 1158 1 %IF (LUB$K_NEG_BLN NEQ ISB$K_NEG_LUB)  
M 1159 1 %THEN  
M 1160 1 |
```

```
: 237 M 1161 1 COMPILETIME
: 238 M 1162 1     VAL1 = -ISBSK_NEG_LUB,
: 239 M 1163 1     VAL2 = -LUBSK_NEG_BLN;
: 240 M 1164 1
: 241 M 1165 1 %ERROR (' LUBSK_NEG_BLN is not equal to ISBSK_NEG_LUB. ')
: 242 M 1166 1     ' This probably means that the LUB has been extended'.
: 243 M 1167 1     ' without editing the ISB to allow for it. Please edit file OTSISB.MDL, making the -F,B,',
: 244 M 1168 1     %NUMBER (VAL1), ' be -F,B,', %NUMBER (VAL2))
: 245 M 1169 1 %FI
: 246 M 1170 1
: 247 M 1171 1 %;
: 248 M 1172 1
: 249 M 1173 1
: 250 M 1174 1 EQUATED SYMBOLS:
: 251 M 1175 1
: 252 M 1176 1
: 253 M 1177 1 LITERAL
: 254 M 1178 1     K_TOTAL_CCB_LEN = LUBSK_LUB_LEN + ISBSK_ISB_LEN + RABSC_BLN;           ! length of LUB+ISB+RAB
: 255 M 1179 1
: 256 M 1180 1
: 257 M 1181 1 PSECT DECLARATIONS:
: 258 M 1182 1
: 259 M 1183 1 DECLARE_PSECTS (OTS);                                ! declare PSECTs for OTSS facility
: 260 M 1184 1
: 261 M 1185 1 OWN STORAGE:
: 262 M 1186 1
: 263 M 1187 1 +
: 264 M 1188 1     The following quadword is the header of the I/O active queue. Items
: 265 M 1189 1     are manipulated on this queue using the INSQUE and REMQUE instructions.
: 266 M 1190 1 -
: 267 M 1191 1
: 268 M 1192 1 OWN
: 269 M 1193 1     OTSSQ_IO_ACTIVE : VECTOR [2];
: 270 M 1194 1
: 271 M 1195 1
: 272 M 1196 1 EXTERNAL REFERENCES:
: 273 M 1197 1
: 274 M 1198 1
: 275 M 1199 1 EXTERNAL ROUTINE
: 276 M 1200 1     LIB$GET_VM,                                     ! Allocate virtual memory
: 277 M 1201 1     LIB$FREE_VM,                                    ! Deallocate virtual memory
: 278 M 1202 1     LIB$STOP : NOVALUE,                            ! Signal a fatal error
: 279 M 1203 1     OTSS$FREE_LUN;                                ! Deallocate a LUN
: 280 M 1204 1
: 281 M 1205 1 EXTERNAL LITERAL
: 282 M 1206 1     OTSS_FATINTERR : UNSIGNED (%BPVAL);          ! condition value for FATAL INTERNAL ERROR
: 283 M 1207 1
: 284 M 1208 1
: 285 M 1209 1 +
: 286 M 1210 1     The following externals represent the global part of the CCB
: 287 M 1211 1     data base.
: 288 M 1212 1 -
: 289 M 1213 1
: 290 M 1214 1 EXTERNAL
: 291 M 1215 1     OTSS$V_CCB_INIT : VOLATILE,                  ! True if INIT done
: 292 M 1216 1     OTSS$AA_LUB_TAB : VOLATILE OTSS$LUB_TAB_ST [-LUBSK_ILUN_MIN + LUBSK_LUN_MAX + T, LUBSK_ILUN_MIN], ! Pointers to CCBs
```

: 294 1218 1 OTSSV\_JOINPROG : VOLATILE BITVECTOR, | True if LUN has I/O active  
: 295 1219 1 OTSSA\_CUR\_LUB, | The current LUB  
: 296 1220 1 OTSSL\_CUR\_LUN, | The current logical unit  
: 297 1221 1 OTSSL\_LVL\_CTR; | -1 = ILDE, 0 = 1 I/O in progress.  
: 298 1222 1  
: 299 1223 1 BUILTIN  
: 300 1224 1 INSQUE, | Insert an item in a queue  
: 301 1225 1 REMQUE, | Remove an item from a queue  
: 302 1226 1 TESTBITSS, | Test bit, set it, return true if it was set.  
: 303 1227 1 TESTBITCC; | Test bit, clear it, return true if it was clear.  
: 304 1228 1  
: 305 1229 1 !<BLF/PAGE>

```
: 307      1230 1 !+
.: 308      1231 1 !+ The following field set represents an item pushed onto the
.: 309      1232 1 !+ I/O Active list. It contains the ISB, the prompt buffer, the
.: 310      1233 1 !+ current size of the prompt buffer, and the timeout value from
.: 311      1234 1 !+ the RAB.
.: 312      1235 1 !-
.: 313      1236 1
.: 314      1237 1 FIELD
.: 315      1238 1   PUSH_ITEM =
.: 316      1239 1     SET
.: 317      1240 1     PUSHSA_NEXT = [0, 0, %BPVAL, 0],           ! Next item
.: 318      1241 1     PUSHSA_PREV = [4, 0, %BPVAL, 0],         Previous item
.: 319      1242 1     PUSHSL_STS = [8, 0, %BPVAL, 0],          RMS status
.: 320      1243 1     PUSHSL_STV = [12, 0, %BPVAL, 0],        RMS extra status
.: 321      1244 1     PUSH$W_LUN = [16, 0, 16, 1],          Logical unit number
.: 322      1245 1     PUSH$B_PSZ = [18, 0, 8, 0],           Prompt buffer size
.: 323      1246 1     PUSH$B_TMO = [19, 0, 8, 0],           The RMS timeout value
.: 324      1247 1     PUSH$V_I0_ACT = [20, 0, 1, 0],        The I/O Active flag
.: 325      1248 1     PUSH$V_FAKE = [20, 1, 1, 0],         The "fake" flag
.: 326      1249 1     PUSH$V_PMT = [20, 2, 1, 0],          Set if there is a prompt buffer.
.: 327      1250 1     PUSH$T_PROMPT = [21, 0, 0, 0],       ! The prompt buffer
.: 328      1251 1     PUSH$X_ISB = [LUB$K_PBUF_SIZ + 21, 0, 0, 0] ! The ISB
.: 329      1252 1     TES;
.: 330      1253 1
.: 331      1254 1 LITERAL
.: 332      1255 1     PUSH$K_LENGTH = 21 + LUB$K_PBUF_SIZ + ISB$K_ISB_LEN; ! Number of bytes to allocate
.: 333      1256 1
```

335 1257 1 ROUTINE INITIALIZE : NOVALUE = ! Set up OWN storage  
336 1258 1  
337 1259 1 ++  
338 1260 1 FUNCTIONAL DESCRIPTION:  
339 1261 1  
340 1262 1 Set up the LUB table, I/O Active queue and OTSS\$L\_CUR\_LUN.  
341 1263 1 The LUB table and I/O active queue must be set up at run time  
342 1264 1 because they must be initialized with addresses, and this  
343 1265 1 cannot be done at link time or they will cease to be position  
344 1266 1 independent. They must be initialized with addresses because  
345 1267 1 they are used by INSQUE and REMQUE to avoid disabling ASTs.  
346 1268 1  
347 1269 1 CALLING SEQUENCE:  
348 1270 1  
349 1271 1 IF (NOT .OTSS\$V\_CCB\_INIT) THEN INITIALIZE ();  
350 1272 1  
351 1273 1 FORMAL PARAMETERS:  
352 1274 1  
353 1275 1  
354 1276 1  
355 1277 1 IMPLICIT INPUTS:  
356 1278 1  
357 1279 1 OTSS\$AA\_LUB\_TAB  
358 1280 1 OTSS\$Q\_I0\_ACTIVE  
359 1281 1 OTSS\$V\_I0INPROG  
360 1282 1 OTSS\$L\_CUR\_LUN  
361 1283 1  
362 1284 1 IMPLICIT OUTPUTS:  
363 1285 1  
364 1286 1 OTSS\$AA\_LUB\_TAB  
365 1287 1 OTSS\$Q\_I0\_ACTIVE  
366 1288 1 OTSS\$V\_I0INPROG  
367 1289 1 OTSS\$L\_CUR\_LUN  
368 1290 1 OTSS\$V\_CCB\_INIT  
369 1291 1  
370 1292 1 SIDE EFFECTS:  
371 1293 1  
372 1294 1  
373 1295 1 --  
374 1296 1  
375 1297 2 BEGIN  
376 1298 2  
377 1299 2 LOCAL  
378 1300 2  
379 1301 2 + The following cell keeps track of whether or not ASTs were disabled  
380 1302 2 when we were called.  
381 1303 2 -  
382 1304 2 AST\_STATUS;  
383 1305 2  
384 1306 2  
385 1307 2 + First disable ASTs. Then, if the initialization has not yet been  
386 1308 2 done, do it. The initialization will have been done if an AST went  
387 1309 2 off between the test of OTSS\$V\_CCB\_INIT and this point.  
388 1310 2 -  
389 1311 2 AST\_STATUS = \$SETAST (ENBFLG = 0);  
390 1312 2  
391 1313 3 IF ( NOT .OTSS\$V\_CCB\_INIT)

```

392      1314 2      THEN
393      1315 2      BEGIN
394      1316 2
395      1317 3      + We must do the initialization. First set the LUB table to be empty.
396      1318 3      Note that LUBs in use by FORTRAN are not touched. FORTRAN leaves the
397      1319 3      first longword non-zero for entries it is using.
398      1320 3
399      1321 3
400      1322 3      INCR LUN FROM LUBSK_LUN_MIN TO LUBSK_LUN_MAX DO
401      1323 3          IF (.OTSS$AA_LUB_TAB [.LUN, 0] EQ[ 0]) THEN
402      1324 3              OTSS$AA_LUB_TAB [.LUN, 0] = OTSS$AA_LUB_TAB [.LUN, 1] = OTSS$AA_LUB_TAB [.LUN, 0];
403      1325 3
404      1326 3
405      1327 3      + Now make the I/O active queue empty.
406      1328 3
407      1329 3          OTSSQ_IO_ACTIVE [0] = OTSSQ_IO_ACTIVE [1] = OTSSQ_IO_ACTIVE [0];
408      1330 3
409      1331 3      + Mark that the initialization has been done, so it won't be done again.
410      1332 3
411      1333 3          OTSS$V_CCB_INIT = 1;
412      1334 2          END;
413      1335 2
414      1336 2
415      1337 2      + If ASTs were enabled at entry, re-enable them.
416      1338 2
417      1339 2
418      1340 2      IF (.AST_STATUS EQ SSS_WASSET) THEN $SETAST (ENBFLG = 1);
419
420      1342 2      RETURN;
421      1343 1      END;

```

! of routine INITIALIZE

```

.TITLE OTSS$CCB
.IDENT \1-057\
.PSECT _OTSS$DATA,NOEXE, PIC.2
00000 OTSSQ_IO_ACTIVE:
.BLKB 8
.EXTRN LIB$GET VM, LIB$FREE VM
.EXTRN LIB$STOP, OTSS$FREE LUN
.EXTRN OTSS$FATINTERR, OTSS$V_CCB_INIT
.EXTRN OTSS$AA_LUB_TAB
.EXTRN OTSS$V_IOPINPROG
.EXTRN OTSS$A_CUR_LUB, OTSS$L_CUR_LUN
.EXTRN OTSS$L_LVL_CTR, SYSS$SETAST
.PSECT _OTSS$CODE,NOWRT, SHR, PIC.2

```

003C 00000 INITIALIZE:

55 00000000G	00 9E 00002	.WORD Save R2,R3,R4,R5
54 00000000G	00 9E 00009	MOVAB SYSS\$SETAST, R5
53 00000000	EF 9E 00010	MOVAB OTSS\$V_CCB_INIT, R4
	7E D4 00017	MOVAB OTSSQ_IO_ACTIVE, R3
65	01 FB 00019	CLRL -(SP)
		CALLS #1, SYSS\$SETAST

1257

1311

	31	64	E8 0001C	BLBS	OTSS\$V_CCB_INIT, 3\$	: 1313
	51	08	CE 0001F	MNEGL	#8_LUN	: 1322
	52	00000000G0041	7E 00022 1\$:	MOVAQ	OTSS\$AA_LUB_TAB+64[LUN], R2	: 1323
		62	D5 0002A	TSTL	(R2)	
		OD	12 0002C	BNEQ	2\$	
		00000000G0041	7F 0002E	PUSHAQ	OTSS\$AA_LUB_TAB+68[LUN]	: 1324
DF	9E	52	D0 00035	MOVL	R2, @(SP)+	
	62	52	D0 00038	MOVL	R2, (R2)	
	51	00000077	8F F3 0003B 2\$:	AOBLEQ	#119_LUN, 1\$	: 1323
	51	63	9E 00043	MOVAB	OTSS\$IO_ACTIVE, R1	: 1329
04	A3	51	D0 00046	MOVL	R1, OTSS\$IO_ACTIVE+4	
	63	51	D0 0004A	MOVL	R1, OTSS\$IO_ACTIVE	
	64	01	D0 0004D	MOVL	#1, OTSS\$V_CCB_INIT	: 1333
	09	50	D1 00050 3\$:	CMPL	AS1_STATUS, #9	: 1340
		05	12 00053	BNEQ	4\$	
		01	DD 00055	PUSHL	#1	
	65	01	FB 00057	CALLS	#1, SYSSSETAST	
		04	0005A 4\$:	RET		: 1343

: Routine Size: 91 bytes. Routine Base: \_OTSS\$CODE + 0000

```
: 423      1344 1 ROUTINE PUSH_FAKE : CALL_CCB =           ! Push a "fake" active record
: 424
: 425
: 426
: 427
: 428
: 429
: 430
: 431
: 432
: 433
: 434
: 435
: 436
: 437
: 438
: 439
: 440
: 441
: 442
: 443
: 444
: 445
: 446
: 447
: 448
: 449
: 450
: 451
: 452
: 453
: 454
: 455
: 456
: 457
: 458
: 459
: 460
: 461
: 462
: 463
: 464
: 465
: 466
: 467
: 468
: 469
: 470
: 471
: 472
: 473
: 474
: 475
: 476
: 477
: 478
: 479
: 480      1345 1
: 481      1346 1 ++
: 482      1347 1 FUNCTIONAL DESCRIPTION:
: 483      1348 1     Push onto the I/O Active queue a place holder. This is to
: 484      1349 1     satisfy POP_ACTIVE when we can't actually push the CCB.
: 485      1350 1
: 486      1351 1
: 487      1352 1 CALLING SEQUENCE:
: 488      1353 1
: 489      1354 1     CALL PUSH_FAKE ();
: 490      1355 1
: 491      1356 1 FORMAL PARAMETERS:
: 492      1357 1     NONE
: 493      1358 1
: 494      1359 1
: 495      1360 1 IMPLICIT INPUTS:
: 496      1361 1
: 497      1362 1     OTSS$Q_IO_ACTIVE
: 498      1363 1     OTSS$L_CUR_LUN
: 499      1364 1
: 500      1365 1 IMPLICIT OUTPUTS:
: 501      1366 1
: 502      1367 1     OTSS$Q_IO_ACTIVE      Holds previous I/O on this LUN
: 503      1368 1
: 504      1369 1 SIDE EFFECTS:
: 505      1370 1
: 506      1371 1     Calls LIB$GET_VM to get virtual memory.
: 507      1372 1
: 508      1373 1
: 509      1374 2 BEGIN
: 510      1375 2
: 511      1376 2 EXTERNAL REGISTER
: 512      1377 2     CCB : REF_BLOCK [, BYTE];
: 513      1378 2
: 514      1379 2 LOCAL
: 515      1380 2
: 516      1381 2 + Declare the pointer to the block to push.
: 517      1382 2 -
: 518      1383 2     PUSH : REF_BLOCK [PUSH$K_LENGTH, BYTE] FIELD (PUSH_ITEM),
: 519      1384 2     LUN;
: 520      1385 2
: 521      1386 2     LUN = .OTSS$L_CUR_LUN;
: 522      1387 2
: 523      1388 2 + Get virtual memory to hold the fake activation record.
: 524      1389 2 -
: 525      1390 3 BEGIN
: 526      1391 3
: 527      1392 3 LOCAL
: 528      1393 3     GET_VM_RESULT;
: 529      1394 3
: 530      1395 3     GET_VM_RESULT = LIB$GET_VM (%REF (PUSH$K_LENGTH), PUSH);
: 531      1396 3
: 532      1397 3     IF ( NOT .GET_VM_RESULT) THEN RETURN (OTSS$K_PUSH_FAIL);
: 533      1398 3
: 534      1399 2 END:
: 535      1400 2 !+
```

```

: 480      1401 2 ! Copy the old LUN into the fake record, and mark it as fake.
: 481      1402 2 !-
: 482      1403 2   PUSH [PUSH$W_LUN] = .LUN;
: 483      1404 2   PUSH [PUSH$V_FAKE] = 1;
: 484      1405 2 +
: 485      1406 2 ! Put this item on the I/O Active list.
: 486      1407 2 !-
: 487      1408 2   INSQUE (.PUSH, OTSSQ_IO_ACTIVE);
: 488      1409 2 +
: 489      1410 2 ! We also set OTSSL_CUR_LUN to LUBSK_LUN_MAX+1 to prevent an
: 490      1411 2 ! AST from pushing that [UB again. An extra push before this point
: 491      1412 2 ! does not cause any harm (only wastes a little time).
: 492      1413 2 !-
: 493      1414 2   OTSSL_CUR_LUN = LUBSK_LUN_MAX + 1;
: 494      1415 2   RETURN (OTSSK_PUSH_OK);
: 495      1416 1   END;                                ! of routine PUSH_FAKE

```

000C 00000 PUSH_FAKE:							
							.WORD
53	00000000G	00	9E	00002	MOVAB	Save R2,R3	1344
5E		08	C2	00009	SUBL2	OTSSL_CUR_LUN, R3	
52		E3	D0	0000C	MOVL	#8, SP	
04	AE	0121	AE	9F 0000F	PUSHAB	OTSSL_CUR_LUN, LUN	1386
		04	8F	3C 00012	MOVZWL	PUSH	1395
00000000G	00		AE	9F 00018	PUSHAB	#289, 4(SP)	
			02	FB 0001B	CALLS	4(SP)	
			04	50 E8 00022	BLBS	#2, LIB\$GET_VM	1397
			50	03 D0 00025	MOVL	GET_VMRRESULT, 1\$	
				04 00028	RET	#3, R0	
10	A0	50	AE	00029 1\$:	MOVL	PUSH, R0	1403
14	A0	04	52	B0 0002D	MOVW	LUN, 16(R0)	
00000000	EF		C2	88 00031	BISB2	#2, 20(R0)	1404
			60	0E 00035	INSQUE	(R0), OTSSQ_IO_ACTIVE	1408
			63	78 8F 9A 0003C	MOVZBL	#120, OTSS\$E_CUR_LUN	1414
			50	01 D0 00040	MOVL	#1, R0	1415
				04 00043	RET		1416

: Routine Size: 68 bytes. Routine Base: \_OTSSCODE + 005B

```
: 497 1417 1 ROUTINE PUSH_ACTIVE (LOGICAL_UNIT,  
: 498 1418 1 RECURSIVE_IO  
: 499 1419 1 ) : CALL_CCB =  
: 500 1420 1  
: 501 1421 1 ++  
: 502 1422 1 FUNCTIONAL DESCRIPTION:  
: 503 1423 1  
: 504 1424 1 Place the ISB, etc. of the currently active logical unit on the  
: 505 1425 1 I/O Active queue so that another I/O statement may be started.  
: 506 1426 1 The I/O statement to be started may be on the same or another  
: 507 1427 1 logical unit as the one being interrupted. When the new I/O  
: 508 1428 1 statement is complete the old one will be continued, so the I/O  
: 509 1429 1 active queue has a first-in-first-out discipline.  
: 510 1430 1  
: 511 1431 1 CALLING SEQUENCE:  
: 512 1432 1  
: 513 1433 1 RESULT = CALL PUSH_ACTIVE (LUN, RECURSIVE_IO);  
: 514 1434 1  
: 515 1435 1 FORMAL PARAMETERS:  
: 516 1436 1  
: 517 1437 1 LOGICAL_UNIT.rl.v The new LUN  
: 518 1438 1 RECURSIVE_IO.rl.v True if this LUN was already active  
: 519 1439 1  
: 520 1440 1 IMPLICIT INPUTS:  
: 521 1441 1  
: 522 1442 1 OTSSAA_LUB_TAB  
: 523 1443 1 OTSSQ_IO_ACTIVE  
: 524 1444 1 OTSSL_CUR_LUN  
: 525 1445 1  
: 526 1446 1 IMPLICIT OUTPUTS:  
: 527 1447 1  
: 528 1448 1 OTSSQ_IO_ACTIVE Holds previous I/O on this LUN  
: 529 1449 1  
: 530 1450 1 SIDE EFFECTS:  
: 531 1451 1  
: 532 1452 1 Calls LIB$GET_VM to get virtual memory.  
: 533 1453 1--  
: 534 1454 1  
: 535 1455 2 BEGIN  
: 536 1456 2  
: 537 1457 2 EXTERNAL REGISTER  
: 538 1458 2 CCB : REF BLOCK [, BYTE];  
: 539 1459 2  
: 540 1460 2 LOCAL  
: 541 1461 2  
: 542 1462 2 Declare the pointer to the block to push.  
: 543 1463 2  
: 544 1464 2 PUSH : REF BLOCK [PUSH$K_LENGTH, BYTE] FIELD (PUSH_ITEM);  
: 545 1465 2  
: 546 1466 2  
: 547 1467 2 If there is no need to push anything, push a fake activation  
: 548 1468 2 record to satisfy POP_ACTIVE.  
: 549 1469 2  
: 550 1470 2  
: 551 1471 2 IF (.OTSSL_CUR_LUN GTR LUBSK_LUN_MAX) THEN RETURN (PUSH_FAKE());  
: 552 1472 2  
: 553 1473 2 !+
```

554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610

1474 2 | Check for this being an AST between the clearing of OTSS\$V IOINPROG  
1475 2 | and the setting of OTSSL\_CUR\_LUN to LUB\$K\_LUN\_MAX + 1. If it  
1476 2 | is we cannot push the CCB since, with RECURSIVE\_I/O clear,  
1477 2 | OTSS\$V\_IOINPROG will be cleared before the call to POP\_ACTIVE.  
1478 2 | and we might try to pop into a deallocated CCB.  
1479 2 |  
1480 2 |  
1481 2 | IF ((.OTSSL\_CUR\_LUN EQ .LOGICAL\_UNIT) AND ( NOT .RECURSIVE\_I/O)) THEN RETURN (PUSH\_FAKE ());  
1482 2 |  
1483 2 | CCB = .OTSS\$AA\_LUB\_TAB [.OTSSL\_CUR\_LUN, 0];  
1484 2 |  
1485 2 | If the queue is empty then the deallocation code has removed the LUB  
1486 2 | from the LUB table but has not yet popped OTSSL\_CUR\_LUN. Since  
1487 2 | the deallocation code will finish its deallocation no matter what  
1488 2 | we do here we need not push anything. If any I/O is tried to this  
1489 2 | LUN it will create a new LUB. The recursive flag may be set  
1490 2 | needlessly, but that will only cause a problem in languages which  
1491 2 | do not support recursive I/O, and, actually, the higher I/O has not  
1492 2 | quite finished yet, so that is OK.  
1493 2 |  
1494 2 |  
1495 2 | IF (.CCB EQA OTSS\$AA\_LUB\_TAB [.OTSSL\_CUR\_LUN, 0]) THEN RETURN (PUSH\_FAKE ());  
1496 2 |  
1497 2 |  
1498 2 | The LUB is still allocated, do some consistency checks.  
1499 2 | We cannot check OTSS\$AA\_CUR\_LUB since we may be in an AST that  
1500 2 | occurred after the update of OTSS\$AA\_CUR\_LUB but before OTSSL\_CUR\_LUN.  
1501 2 |  
1502 2 | CCB = .CCB + (.CCB - CCB [LUB\$Q\_QUEUE]);  
1503 2 |  
1504 2 | IF (.CCB [LUB\$W\_LUN] NEQ .OTSSL\_CUR\_LUN) THEN LIB\$STOP (OTSS\_FATINTERR);  
1505 2 |  
1506 2 |  
1507 2 | Get virtual memory to hold the old ISB, etc.  
1508 2 |  
1509 3 | BEGIN  
1510 3 |  
1511 3 | LOCAL  
1512 3 | GET\_VM\_RESULT;  
1513 3 |  
1514 3 | GET\_VM\_RESULT = LIB\$GET\_VM (%REF (PUSH\$K\_LENGTH), PUSH);  
1515 3 |  
1516 3 | IF ( NOT .GET\_VM\_RESULT) THEN RETURN (OTSS\$K\_PUSH\_FAIL);  
1517 3 |  
1518 2 | END;  
1519 2 |  
1520 2 | Make sure there is no RMS I/O active on the RAB.  
1521 2 |  
1522 2 |  
1523 2 | IF (.RECURSIVE\_I/O) THEN SWAIT (RAB = .CCB);  
1524 2 |  
1525 2 |  
1526 2 | Copy the ISB and a few other things that need to be preserved  
1527 2 | over recursive I/O into the block we just allocated.  
1528 2 |  
1529 2 | CH\$MOVE (ISB\$K\_ISB\_LEN, .CCB - ISB\$K\_ISB\_LEN - LUB\$K\_LUB\_LEN, PUSH [PUSH\$X\_ISB]);  
1530 2 | PUSH [PUSH\$V\_PMT] = .CCB [RAB\$V\_PMT];

```

611      1531 2
612      1532 3
613      1533 2
614      1534 3
615      1535 3
616      1536 3
617      1537 2
618      1538 2
619      1539 2
620      1540 2
621      1541 2
622      1542 2
623      1543 2
624      1544 2
625      1545 2
626      1546 2
627      1547 2
628      1548 2
629      1549 2
630      1550 2
631      1551 2
632      1552 2
633      1553 2
634      1554 2
635      1555 2
636      1556 2
637      1557 2
638      1558 2
639      1559 2
640      1560 2
641      1561 2
642      1562 2
643      1563 2
644      1564 1

       IF (.PUSH [PUSH$V_PMT])
       THEN
         BEGIN
           CH$MOVE (.CCB[RAB$B_PSZ], .CCB[RAB$L_PBF], PUSH[PUSH$T_PROMPT]);
           PUSH[PUSH$B_PSZ] = .CCB[RAB$B_PSZ];
           END;

           PUSH[PUSH$B_TMO] = .CCB[RAB$B_TMO];
           PUSH[PUSH$L_STS] = .CCB[RAB$L_STS];
           PUSH[PUSH$L_STV] = .CCB[RAB$L_STV];
           PUSH[PUSH$V_IO_ACT] = .CCB[LUB$V_IO_ACTIVE];
           PUSH[PUSH$V_FAKE] = 0;

           + Record the logical unit number so that POP_ACTIVE knows where to
             restore this item when it is popped.
           - PUSH[PUSH$W_LUN] = .CCB[LUB$W_LUN];
           + Put this item on the I/O Active list.
           - INSQUE(.PUSH, OTSSQ_IO_ACTIVE);
           + That LUB is no longer the active one, mark it so.
           - CCB[LUB$V_IO_ACTIVE] = 0;
           + We also set OTSSL_CUR_LUN to LUB$K_LUN_MAX+1 to prevent an
             AST from pushing that LUN again. An extra push before this point
             does not cause any harm (only wastes a little time).
           - OTSSL_CUR_LUN = LUB$K_LUN_MAX + 1;
           RETURN(OTSSK_PUSH_OK);
           END;

```

! of routine PUSH\_ACTIVE

#### .EXTRN SY\$SWAIT

00FC 00000 PUSH_ACTIVE:					
				WORD	Save R2,R3,R4,R5,R6,R7
		57 0000000G	00 9E 00002	MOVAB	OTSSL_CUR_LUN, R7
00000077	5E	8F	08 C2 00009	SUBL2	#8, SP
			67 D1 0000C	CMPL	OTSSL_CUR_LUN, #119
	04	AC	1D 14 00013	BGTR	2\$
			67 D1 00015	CMPL	OTSSL_CUR_LUN, LOGICAL_UNIT
			04 12 00019	BNEQ	1\$
	13	08	AC E9 0001B	BLBC	RECURSIVE IO, 2\$
	50	50	67 D0 0001F	1\$:	MOVL
		00000000G0040	7E 00022	MOVAQ	OTSSL_CUR_LUN, R0
			60 D0 0002A	MOVL	OTSSAA_LUB_TAB+64[R0], R0
		5B	5B D1 0002D	(R0), CCB	
		50	05 12 00030	CMPL	CCB, R0
	86	AF	00 FB 00032	2\$:	BNEQ
			04 00036	CALLS	3\$, PUSH_FAKE
50	5B	5B	C3 00037	3\$:	RET
		A04B	9E 0003B	SUBL3	CCB, CCB, R0
				MOVAB	88(R0)[CCB], CCB

67	C6	AB	10		00	EC	00040	CMPV	#0, #16, -58(CCB), OTSSL_CUR_LUN	: 1504
				00000000G	00	0D	13	00046	BEQL	48
				00000000G	00	8F	DD	00048	PUSHL	#OTSS_FATINTERR
			04	AE	01	FB	0004E	CALLS	#1, LIBSTOP	
			04	AE	AE	9F	00055	48:	PUSHAB	PU\$H
			04	AE	8F	3C	00058	MOVZWL	#289, 4(SP)	
			04	AE	AE	9F	0005E	PUSHAB	4(SP)	
			04	AE	02	FB	00061	CALLS	#2, LIB\$GET_VM	
			50		50	E8	00068	BLBS	GET_VM_RESULT, 5\$	
			50		03	DO	0006B	MOVL	#3, R0	
					04	04	0006E	RET		
					09	AC	E9	0006F	BLBC	RECURSIVE_IO, 6\$
				00000000G	00	5B	DD	00073	PUSHL	CCB
				00000000G	00	01	FB	00075	CALLS	#1, SYS\$WAIT
			56	04	AE	D0	0007C	68:	MOVL	PU\$H, R6
			01	04	8F	28	00080	MOV3	#188, -288(CCB), 101(R6)	
			02	04	06	EF	00089	EXTZV	#6, #1, 7(CCB), R0	
			02	14	50	F0	0008F	INSV	R0, #2, #1, 20(R6)	
			14	A6	02	E1	00095	BBC	#2, 20(R6), 7\$	
			50	34	AB	9A	0009A	MOVZBL	52(CCB), R0	
			34	A6	50	28	0009E	MOV3	R0, 248(CCB), 21(R6)	
			34	A6	AB	90	000A4	MOVB	52(CCB), 18(R6)	
			1F	A6	1F	AB	90	000A9	MOVB	31(CCB), 19(R6)
			08	A6	08	AB	70	000AE	MOVG	8(CCB), 8(R6)
			01	01	01	EF	000B3	EXTZV	#1, #1, -4(CCB), R0	
			00	00	50	F0	000B9	INSV	R0, #0, #1, 20(R6)	
			14	A6	02	8A	000BF	BICB2	#2, 20(R6)	
			'0	A6	AB	B0	000C3	MOVW	-58(CCB), 16(R6)	
			EF	EF	66	OE	000C8	INSQUE	(R6), OTSSQ_IO_ACTIVE	
			FC	AB	02	8A	000CF	BICB2	#2, -4(CCB)	
			67		78	8F	9A	000D3	MOVZBL	#120, OTSSL_CUR_LUN
			50		01	DO	000D7	MOVL	#1, R0	
					04	04	000DA	RET		

; Routine Size: 219 bytes, Routine Base: \_OTSSCODE + 009F

```
: 646    1565 1 ROUTINE ALLOCATE (LOGICAL_UNIT) : CALL_CCB = ! Allocate LUB/ISB/RAB
: 647    1566 1
: 648    1567 1 ++
: 649    1568 1 FUNCTIONAL DESCRIPTION:
: 650    1569 1
: 651    1570 1     Allocate the LUB/ISB/RAB for this logical unit, watching out for
: 652    1571 1     ASTs which may do the allocation as we are running.
: 653    1572 1
: 654    1573 1 CALLING SEQUENCE:
: 655    1574 1
: 656    1575 1     CALL ALLOCATE (.LOGICAL_UNIT)
: 657    1576 1
: 658    1577 1 FORMAL PARAMETERS:
: 659    1578 1
: 660    1579 1     LOGICAL_UNIT.rl.v      The logical unit number for this CCB
: 661    1580 1
: 662    1581 1 IMPLICIT INPUTS:
: 663    1582 1
: 664    1583 1     OTSS$AA_LUB_TAB
: 665    1584 1
: 666    1585 1 IMPLICIT OUTPUTS:
: 667    1586 1
: 668    1587 1     OTSS$AA_LUB_TAB
: 669    1588 1     CCB
: 670    1589 1
: 671    1590 1 SIDE EFFECTS:
: 672    1591 1
: 673    1592 1     Calls LIB$GET_VM to get virtual memory.
: 674    1593 1     May call LIB$FREE_VM to free that same virtual memory.
: 675    1594 1 --
: 676    1595 1
: 677    1596 2 BEGIN
: 678    1597 2
: 679    1598 2 EXTERNAL REGISTER
: 680    1599 2     CCB : REF BLOCK [, BYTE];
: 681    1600 2
: 682    1601 2 LOCAL
: 683    1602 2     INSQUE_ADDR,          ! Address for INSQUE instruction
: 684    1603 2     REMQUE_ADDR,        ! Address for REMQUE instruction
: 685    1604 2     CCB_ADDR;           ! Address of the allocated CCB
: 686    1605 2
: 687    1606 2 ++
: 688    1607 2     Test the definitions of the LUB and ISB for consistency. This is
: 689    1608 2     purely a compile-time test; it generates no code.
: 690    1609 2 -
: 691    1610 2     TEST_LUB_ISB;
: 692    1611 2 ++
: 693    1612 2     We must allocate. This case is a little complex since an AST may
: 694    1613 2     allocate the LUB. We handle this by preparing the LUB and then
: 695    1614 2     checking to see if an AST allocated one. If so, we deallocate ours.
: 696    1615 2 -
: 697    1616 3 BEGIN
: 698    1617 3
: 699    1618 3 LOCAL
: 700    1619 3     GET_VM_RESULT;
: 701    1620 3
: 702    1621 3     GET_VM_RESULT = LIB$GET_VM (%REF (K_TOTAL_CCB_LEN), CCB_ADDR);
```

```
: 703      1622 3     IF ( NOT .GET_VM_RESULT) THEN RETURN (OTSSK_PUSH_FAIL);
: 704
: 705
: 706
: 707
: 708
: 709
: 710
: 711
: 712
: 713
: 714
: 715      1623 3
: 716      1624 3     END;
: 717      1625 2
: 718      1626 2     + Clear the newly allocated LUN and RAB (but not ISB). Adjust the
: 719      1627 2     contents of the control block pointer (CCB) so that it points to
: 720      1628 2     the beginning of the RAB. (The ISB and LUB precede the RAB using
: 721      1629 2     negative offsets with respect to register CCB.)
: 722      1630 2     Set the unit number in the newly allocated LUB.
: 723      1631 2
: 724      1632 2
: 725      1633 2
: 726      1634 2     [CB = .CCB_ADDR;
: 727      1635 2     CHSFILL (0, LUB$K_LUB_LEN + RAB$C_BLN, .CCB + ISBSK_ISB_LEN);
: 728      1636 2     [CB = .CB + ISBSR_ISB_LEN + LUB$R_LUB_LEN;
: 729      1637 2     [CB [LUBSW_LUN] = .LOGICAL_UNIT;
: 730      1638 2
: 731      1639 2     + Initialize RAB to constants which never change.
: 732      1640 2     Block ID, block length, and bit to make $PUT do $UPDATE if
: 733      1641 2     record exists. Also truncate on sequential $PUT not at EOF.
: 734      1642 2     Note: TPT bit depends on FOP TRN bit being set in order to take effect.
: 735      1643 2     Set read-ahead, write-behind and locate mode for GETs.
: 736      1644 2
: 737      1645 2     [CB [RAB$B_BID] = RAB$C_BID;
: 738      1646 2     [CB [RAB$B_BLN] = RAB$C_BLN;
: 739      1647 2     [CB [RAB$V_UIF] = 1;
: 740      1648 2     [CB [RAB$V_TPT] = 1;
: 741      1649 2     [CB [RAB$V_RAH] = 1;
: 742      1650 2     [CB [RAB$V_WBH] = 1;
: 743      1651 2     [CB [RAB$V_LOC] = 1;
: 744      1652 2
: 745      1653 2     + Set up LUBSA_BUDDY_PTR. If this CCB is not its own buddy, this
: 746      1654 2     field will be changed during open.
: 747      1655 2
: 748      1656 2     [CB [LUBSA_BUDDY_PTR] = .CCB;
: 749      1657 2
: 750      1658 2     See if an AST has allocated this LUB/RAB/ISB while we were preparing
: 751      1659 2     ourselves above. If so, we use the allocated one. If the LUB was
: 752      1660 2     allocated by an AST it cannot have I/O active, since the AST must
: 753      1661 2     complete any I/O it starts. In spite of this, it cannot be
: 754      1662 2     deallocated because we have OTSS$V_IOINPROG set for the LUN.
: 755      1663 2     INSQUE_ADDR = OTSS$AA_LUB_TAB [.LOGICAL_UNIT, 1];
: 756      1664 2
: 757      1665 3     IF ( NOT INSQUE ([CB [LUB$Q_QUEUE], ..INSQUE_ADDR))
: 758      1666 2     THEN
: 759      1667 3     BEGIN
: 760      1668 2
: 761      1669 3     This CCB is not the first in the queue, which means that an AST
: 762      1670 3     has allocated one and put it in the queue before us. Remove ours
: 763      1671 3     and deallocate it. We will use the LUB previously on the queue.
: 764      1672 2
: 765      1673 3     REMQUE_ADDR = OTSS$AA_LUB_TAB [.LOGICAL_UNIT, 1];
: 766      1674 2
: 767      1675 3     CASE (REMQUE(..REMQUE_ADDR, (CB)) FROM 0 TO 3 OF
: 768      1676 2     SET
: 769      1677 3
: 770      1678 3     [2] :
```

```
: 760      1679    3    !+ Somebody removed the other entry. This should never happen.  
: 761      1680    3    | -  
: 762      1681    3    LIB$STOP (OTSS_FATINTERR);  
: 763      1682    3       [3] :  
: 764      1683    3    | + The queue was empty. This is unreasonable because OTSS$V_IOINPROG is set.  
: 765      1684    3    | -  
: 766      1685    3    LIB$STOP (OTSS_FATINTERR);  
: 767      1686    3       [0] :  
: 768      1687    3    | +  
: 769      1688    3    All is well. We can now free the CCB we just removed.  
: 770      1689    3    | - It had better be the one we allocated.  
: 771      1690    3       [0] :  
: 772      1691    3    | +  
: 773      1692    3    IF ((.CCB + (.CCB - CCB [LUB$Q_QUEUE])) NEQA (.CCB_ADDR + ISBK_ISB_LEN + LUBSK_LUB_LEN))  
: 774      1693    3    THEN  
: 775      1694    3    LIB$STOP (OTSS_FATINTERR);  
: 776      1695    3       [INRANGE, OUTRANGE] :  
: 777      1696    4    | +  
: 778      1697    3    This should never happen; the only possible values from the REMQUE  
: 779      1698    3    function are 0, 2 and 3.  
: 780      1699    3    | -  
: 781      1700    3    LIB$STOP (OTSS_FATINTERR);  
: 782      1701    3    TES:  
: 783      1702    3       [INRANGE, OUTRANGE] :  
: 784      1703    3    | +  
: 785      1704    3    Now free the LUB we allocated.  
: 786      1705    3    | -  
: 787      1706    3    BEGIN  
: 788      1707    3    LOCAL  
: 789      1708    3       FREE_VM_STATUS;  
: 790      1709    3    | -  
: 791      1710    3    FREE_VM_STATUS = LIB$FREE_VM (%REF (K_TOTAL_CCB_LEN), CCB_ADDR);  
: 792      1711    4    | -  
: 793      1712    4    IF ( NOT .FREE_VM_STATUS) THEN LIB$STOP (OTSS_FATINTERR);  
: 794      1713    4    | -  
: 795      1714    4    END;  
: 796      1715    4    | -  
: 797      1716    4    Now fetch the CCB address. It must still be there because of  
: 798      1717    4    OTSS$V_IOINPROG.  
: 799      1718    4    | -  
: 800      1719    4    CCB = .OTSSA_LUB_TAB [.LOGICAL_UNIT, 0];  
: 801      1720    3    | -  
: 802      1721    3    CCB = .CCB + (.CCB - CCB [LUB$Q_QUEUE]);  
: 803      1722    3    | -  
: 804      1723    3    END;  
: 805      1724    3    | -  
: 806      1725    3    RETURN (OTSSK_PUSH_OK);  
: 807      1726    3    | -  
: 808      1727    2    END;  
: 809      1728    2    | -  
: 810      1729    2    ! of routine ALLOCATE  
: 811      1730    1    END;
```

01FC 00000 ALLOCATE:

```

      58 00000000G 00 9E 00002 .WORD Save R2, R3, R4, R5, R6, R7, R8 : 1565
      57 00000000G 8F D0 00009 MOVAB LIB$STOP R8
      56 00000000G 00 9E 00010 MOVL #NOTSS FA$INTERR, R7
      SE 08 C2 00017 MOVAB OTSSAA_LUB_TAB+68, R6
      04 AE 0164 04 AE 9F 0001A SUBL2 #8, SP
      04 AE 0164 04 AE 9F 00023 PUSHAB CCB_ADDR
      00 00000000G 00 02 FB 00026 MOVZWL #358, 4(SP)
      C4 50 E8 0002D PUSHAB 4(SP)
      50 03 D0 00030 CALLS #2, LIB$GET_VM
      00 04 04 04 00033 BLBS GE$-VM_RESULT, 1$
      00A8 8F 00 00034 1$: MOVL #3, R0
      00 00 00038 RET
      00 00 0003F MOVL CCB_ADDR, CCB
      00 00 00042 MOVAB #0, (SP), #0, #168, 188(CCB) : 1633
      00 00 00047 MOVW 288(R11), CCB : 1635
      C6 AB 0120 CB 9E 00047 MOVW LOGICAL_UNIT, -58(CCB) : 1636
      68 4401 8F B0 0004C MOVW #17409, -(CCB) : 1644
      50 04 AB 9E 00051 MOVAB 4(CCB), R0 : 1646
      B8 AB 00010612 8F C8 00055 BISL2 #67090, (R0) : 1650
      50 04 AC D0 0005C MOVL CCB, -72(CCB) : 1655
      50 04 AC D0 00060 MOVL LOGICAL_UNIT, R0 : 1663
      50 04 6640 7E 00064 MOVAQ OTSSAA_LUB_TAB+68[R0], INSQUE_ADDR
      00 80 A8 AB 0E 00068 INSQUE -88(CCB), @0(INSQUE_ADDR) : 1665
      50 04 6C 13 0006D BEQL 7$
      50 04 AC D0 0006F MOVL LOGICAL_UNIT, R0 : 1673
      51 6640 7E 00073 MOVAQ OTSSAA_LUB_TAB+68[R0], REMQUE_ADDR
      58 00 B1 0F 00077 REMQUE @0(REMQUE_ADDR), CCB : 1675
      50 00 50 DC 0007B MOVPSL R0
      50 02 01 EF 0007D EXTZV #1, #2, R0, R0
      00 00 50 CF 00082 CASEL R0, #0, #3
      0021 0021 000A 00086 2$: .WORD 3$-2$,-
      0021 0021 000A 00086 2$: .WORD 4$-2$,-
      0021 0021 000A 00086 2$: .WORD 4$-2$,-
      0021 0021 000A 00086 2$: .WORD 4$-2$,-

      50 58 17 11 0008E BRB 4$ : 1688
      51 58 C3 00090 3$: SUBL3 CCB, [CCB, R0] : 1696
      50 04 AE 0000120 A04B 9E 00094 MOVAB 88(R0)[CCB], R1
      50 05 8F C1 00099 ADDL3 #288, [CCB_ADDR, R0]
      51 D1 000A2 CMPL R1, R0
      04 AE 0164 05 13 000A5 BEQL 5$ : 1698
      68 68 57 DD 000A7 4$: PUSHL R7
      04 AE 0164 01 FB 000A9 CALLS #1, LIB$STOP
      04 AE 0164 04 AE 9F 000AC 5$: PUSHAB CCB_ADDR
      04 AE 0164 04 AE 9F 000B5 PUSHAB #358, 4(SP)
      00000000G 00 02 FB 000B8 CALLS #2, LIB$FREE_VM
      05 05 50 E8 000BF BLBS FREE_VM_STATUS, 6$ : 1718
      57 68 57 DD 000C2 PUSHL R7
      50 68 01 FB 000C4 CALLS #1, LIB$STOP
      50 04 AC D0 000C7 6$: MOVL LOGICAL_UNIT, R0
      FC A640 7F 000CB PUSHAQ OTSSAA_LUB_TAB+64[R0]
      58 58 9E D0 000CF MOVL @0(SP)+[CCB-
      58 58 C3 000D2 SUBL3 CCB, [CCB, R0] : 1726
      58 58 A04B 9E 000D6 MOVAB 88(R0)[CCB], CCB
      50 01 D0 000DB 7$: MOVL #1, R0 : 1729
    
```

OTSS\$CCB  
1-057

K 7  
16-Sep-1984 01:22:30  
14-Sep-1984 12:39:38

VAX-11 Bliss-32 V4.0-742  
[LIBRTL.SRC]OTSCCB.B32;1

Page 22  
(7)

OTS  
1-C

04 000DE RET

; 1730

; Routine Size: 223 bytes. Routine Base: \_OTSS\$CODE + 017A

; 812 1731 1

; F

; 1

```

: 814      1732 1 GLOBAL ROUTINE OTSS$PUSH_CCB (
: 815          1733 1     LOGICAL_UNIT
: 816          1734 1 ) : JSR_CCB_PUSH =
: 817          1735 1
: 818          1736 1 !++
: 819          1737 1 FUNCTIONAL DESCRIPTION:
: 820          1738 1
: 821          1739 1 Load register CCB with a pointer to the LUB/ISB/RAB for this
: 822          1740 1 logical unit. If no LUB has been allocated, allocate one.
: 823          1741 1 If there is already I/O active push down the old ISB, etc.
: 824          1742 1 POP_ACTIVE will restore it. We already know that this LUN is
: 825          1743 1 not in use by FORTRAN.
: 826          1744 1
: 827          1745 1 CALLING SEQUENCE:
: 828          1746 1
: 829          1747 1     CALL OTSS$PUSH_CCB (logical_unit.rl.v)
: 830          1748 1
: 831          1749 1 FORMAL PARAMETERS:
: 832          1750 1
: 833          1751 1     logical_unit.rl.v      Logical unit - identifies CCB
: 834          1752 1
: 835          1753 1 IMPLICIT INPUTS:
: 836          1754 1
: 837          1755 1     OTSS$V_CCB_INIT
: 838          1756 1     OTSS$AA_LUB_TAB
: 839          1757 1     OTSS$Q_IO_ACTIVE
: 840          1758 1
: 841          1759 1 IMPLICIT OUTPUTS:
: 842          1760 1
: 843          1761 1     CCB                      Set to adr. of allocated LUB/ISB/RAB
: 844          1762 1     OTSS$Q_IO_ACTIVE      Holds previous I/O on this LUN
: 845          1763 1     OTSS$AA_LUB_TAB      Set of adr. of allocated LUB/ISB/RAB
: 846          1764 1                              for logical unit
: 847          1765 1     LUB$W_LUN              Set to logical unit
: 848          1766 1     LUB$V_IO_ACTIVE      Set to indicate active I/O
: 849          1767 1     OTSS$V_CCB_INIT      Always set to 1.
: 850          1768 1
: 851          1769 1 SIDE EFFECTS:
: 852          1770 1
: 853          1771 1     May call LIB$GET_VM to get virtual memory.
: 854          1772 1     In unusual cases, may call LIB$FREE VM to free virtual memory.
: 855          1773 1     The first time entered, calls INITIALIZE, which disables ASTs.
: 856          1774 1 --
: 857          1775 1
: 858          1776 2 BEGIN
: 859          1777 2
: 860          1778 2 EXTERNAL REGISTER
: 861          1779 2     CCB : REF BLOCK [, BYTE];
: 862          1780 2
: 863          1781 2 LOCAL
: 864          1782 2     RECURSIVE_I0;              ! =1 if we are doing recursive I/O
: 865          1783 2
: 866          1784 2 +
: 867          1785 2     If this is the first entry, call INITIALIZE to set up OWN storage.
: 868          1786 2     Note that PUSH_CCB must be entered before POP_CCB, so this is the
: 869          1787 2     first reference to this data base, except for FORTRAN, which is checked
: 870          1788 2     for in INITIALIZE.

```

```
: 871      1789 2 !-
: 872      1790 2
: 873      1791 2   IF ( NOT .OTSS$V_CCB_INIT) THEN INITIALIZE ();
: 874      1792 2
: 875      1793 2   +
: 876      1794 2   Count the level counter. This must be done before the OTSS$V_I0INPROG
: 877      1795 2   bit is set, otherwise an AST could find the OTSS$V_I0INPROG bit set but
: 878      1796 2   level counter -1 which would mean that the PUSH and POP routines
: 879      1797 2   would not be called and OTSS$V_I0INPROG would get cleared by the AST.
: 880      1798 2
: 881      1799 2   OTSSL_LVL_CTR = .OTSSL_LVL_CTR + 1;
: 882      1800 2
: 883      1801 2   +
: 884      1802 2   Mark that this LUN has I/O active so that its LUB (if it has one yet)
: 885      1803 2   will not be deallocated. If it was already active, remember that.
: 886      1804 2   RECURSIVE_IO = (TESTBITSS (OTSS$V_I0INPROG [.LOGICAL_UNIT - LUB$K_ILUN_MIN]));
: 887      1805 2
: 888      1806 2   +
: 889      1807 2   If I/O is currently active, push the presently active unit.
: 890      1808 2
: 891      1809 3   IF (.OTSSL_LVL_CTR NEQ 0)
: 892      1810 2   THEN
: 893      1811 3   BEGIN
: 894      1812 3
: 895      1813 3   LOCAL
: 896      1814 3   PUSH_RESULT;
: 897      1815 3
: 898      1816 3   PUSH_RESULT = PUSH_ACTIVE (.LOGICAL_UNIT, .RECURSIVE_IO);
: 899      1817 3
: 900      1818 3   IF (.PUSH_RESULT NEQ OTSS$K_PUSH_OK) THEN RETURN (.PUSH_RESULT);
: 901      1819 3
: 902      1820 2   END;
: 903      1821 2
: 904      1822 2   +
: 905      1823 2   Allocate the LUB/ISB/RAB if necessary. If an AST allocates it we
: 906      1824 2   must release ours. Note that, because OTSS$V_I0INPROG is set, if an
: 907      1825 2   AST allocates the LUB it will not be deallocated.
: 908      1826 2
: 909      1827 2   CCB = .OTSS$AA_LUB_TAB [.LOGICAL_UNIT, 0];
: 910      1828 2
: 911      1829 3   IF (.CCB NEQA OTSS$AA_LUB_TAB [.LOGICAL_UNIT, 0])
: 912      1830 2   THEN
: 913      1831 3   BEGIN
: 914      1832 3
: 915      1833 3   The CCB is already allocated. Adjust register CCB to point to it.
: 916      1834 3
: 917      1835 3   CCB = .CCB + (.CCB - [CCB LUB$Q_QUEUE]);
: 918      1836 3
: 919      1837 2   END
: 920      1838 3   ELSE
: 921      1839 3   BEGIN
: 922      1840 3
: 923      1841 3   LOCAL
: 924      1842 3   ALLOCATE_RESULT;
: 925      1843 3
: 926      1844 3   ALLOCATE_RESULT = ALLOCATE (.LOGICAL_UNIT);
: 927      1845 3
:           1846 3   IF (.ALLOCATE_RESULT NEQ OTSS$K_PUSH_OK) THEN RETURN (.ALLOCATE_RESULT);
```

```

: 928      1846 3
: 929      1847 2      END:
: 930      1848 2
: 931      1849 2      +
: 932      1850 2      Set OTSSL_CUR_LUN to be the current logical unit number. This is
: 933      1851 2      the cell that controls pushing.
: 934      1852 2      -
: 935      1853 2      OTSSL_CUR_LUN = .LOGICAL_UNIT;
: 936      1854 2      +
: 937      1855 2      Mark this LUB as being the active one, and, if it is participating
: 938      1856 2      in recursive I/O, mark that, too.
: 939      1857 2      -
: 940      1858 2      CCB [LUB$V_IO_ACTIVE] = 1;
: 941      1859 2      CCB [ISBV_RECURSIVE] = .RECURSIVE_IO;
: 942      1860 2      +
: 943      1861 2      Set OTSSA_CUR_LUB to point to the new current LUB.
: 944      1862 2      -
: 945      1863 2      OTSSA_CUR_LUB = .CCB;
: 946      1864 2      +
: 947      1865 2      Initialize the STTM_STAT field of the ISB. We clear these bits so
: 948      1866 2      that the initialization routines at UDF and REC levels can set them
: 949      1867 2      if necessary (unusual) or do nothing to have them cleared.
: 950      1868 2      -
: 951      1869 2      CCB [ISBV_P FORM CH] = 0;
: 952      1870 2      CCB [ISBV_DOLLAR] = 0;
: 953      1871 2      CCB [ISBV_USER ELEM] = 0;
: 954      1872 2      CCB [ISBV_SLASA] = 0;
: 955      1873 2      CCB [ISBV_LAST REC] = 0;
: 956      1874 2      CCB [ISBV_DE ENCODE] = 0;
: 957      1875 2      CCB [ISBV_LIS_HEAP] = 0;
: 958      1876 2      +
: 959      1877 2      When we set OTSSV_IOINPROG we tested it to see if I/O was already active
: 960      1878 2      on this LUN. If it was we must return this information to our
: 961      1879 2      caller because some languages do not permit recursive I/O.
: 962      1880 2      -
: 963      1881 2
: 964      1882 2      IF (.RECURSIVE_IO) THEN RETURN (OTSK_PUSH_ACT);
: 965      1883 2
: 966      1884 2      RETURN (OTSK_PUSH_OK);
: 967      1885 1      END;                                ! End of routine OTSS$PUSH_CCB

```

		52 DD 00000 OTSS\$PUSH_CCB::		
	SE	04 C2 00002	PUSHL	R2
	05 0000000G	52 DD 00005	SUBL2	#4, SP
FD94	CF 0000000G	00 E8 00007	PUSHL	R2
	00 0000000G	00 FB 0000E	BLBS	OTSSV_CCB_INIT, 1\$
	6E	00 D6 00013 1\$:	CALLS	#0, INITIAIZE
52	02 0000000G 00	08 C1 00019	INCL	OTSSL_LVL_CTR
	04 AE	50 D4 0001D	ADDL3	#8, LOGICAL_UNIT, R2
		52 E3 0001F	CLRL	R0
		50 D6 00027	BBCS	R2, OTSSV_IOINPROG, 2\$
		50 D0 00029 2\$:	INCL	R0
			MOVL	R0, RECURSIVE_IO

1732

1791

1799

1804

		00000000G	00	D5	0002D	TSTL	OTSSSL_LVL_CTR	: 1809
			10	13	00033	BEQL	3\$	
			04	AE	DD 00035	PUSHL	RECURSIVE IO	: 1816
			04	AE	DD 00038	PUSHL	LOGICAL UNIT	
	FE06	CF	02	FB	0003B	CALLS	#2, PUSA_ACTIVE	:
		01	50	D1	00040	CMPL	PUSH_RESULT, #1	: 1818
			52	12	00043	BNEQ	7\$	:
		50	00000000G	0042	7E 00045	MOVAQ	OTSSAA_LUB_TAB[R2], R0	: 1827
			5B	60	0004D	MOVL	(R0), CCB	
		50		5B	D1 00050	CMPL	CCB, R0	: 1829
				0B	13 00053	BEQL	4\$	
	50	5B	5B	C3	00055	SUBL3	CCB, CCB, R0	: 1835
		5B	58	A04B	9E 00059	MOVAB	88(R0)[CCB], CCB	
				OC	11 0005E	BRB	5\$	: 1829
				6E	DD 00060	4\$: PUSHL	LOGICAL UNIT	: 1843
	FEBA	CF	01	FB	00062	CALLS	#1, ALLOCATE	
		01	50	D1	00067	CMPL	ALLOCATE_RESULT, #1	: 1845
				2B	12 0006A	BNEQ	7\$	
		00000000G	00	6E	DD 0006C	MOVL	LOGICAL UNIT, OTSSSL_CUR_LUN	: 1853
		FC	AB	02	88 00073	BISB2	#2, -4(CCB)	: 1858
		50	96	AB	9E 00077	MOVAB	-106(CCB), R0	: 1859
	01 A0	01	00	04	AE F0 0007B	INSV	RECURSIVE_IO, #0, #1, 1(R0)	
		00000000G	00	5B	DD 00082	MOVL	CCB, OTSSAA_CUR_LUB	: 1863
				60	94 00089	CLRB	(R0)	: 1875
		05	04	AE	E9 0008B	BLBC	RECURSIVE_IO, 6\$	: 1882
		50		02	DD 0008F	MOVL	#2, R0	
				03	11 00092	BRB	7\$	
		50		01	DD 00094	6\$: MOVL	#1, R0	: 1884
		5E		08	CO 00097	7\$: ADDL2	#8, SP	
				04	BA 0009A	POPR	#^M<R2>	: 1885
				05	0009C	RSB		

: Routine Size: 157 bytes, Routine Base: \_OTSSCODE + 0259

: 968 1886 1

```
: 970      1887 1 ROUTINE DEALLOCATE : CALL_CCB NOVALUE =           ! Deallocate LUB/ISB/RAB
: 971      1888 1
: 972      1889 1 +++
: 973      1890 1 FUNCTIONAL DESCRIPTION:
: 974      1891 1
: 975      1892 1     Deallocate the LUB/ISB/RAB for this logical unit, including
: 976      1893 1     the allocated structures attached to it. Also, deallocate the
: 977      1894 1     LUN.
: 978      1895 1
: 979      1896 1 CALLING SEQUENCE:
: 980      1897 1     CALL DEALLOCATE ()
: 981      1898 1
: 982      1899 1 FORMAL PARAMETERS:
: 983      1900 1
: 984      1901 1     NONE
: 985      1902 1
: 986      1903 1
: 987      1904 1 IMPLICIT INPUTS:
: 988      1905 1
: 989      1906 1     OTSS$AA_LUB_TAB
: 990      1907 1     CCB
: 991      1908 1
: 992      1909 1 IMPLICIT OUTPUTS:
: 993      1910 1
: 994      1911 1     OTSS$AA_LUB_TAB
: 995      1912 1     CCB
: 996      1913 1
: 997      1914 1 SIDE EFFECTS:
: 998      1915 1
: 999      1916 1     Calls LIB$FREE_VM to free virtual memory.
: 1000     1917 1 --
: 1001     1918 1
: 1002     1919 2 BEGIN
: 1003     1920 2
: 1004     1921 2 EXTERNAL REGISTER
: 1005     1922 2     CCB : REF_BLOCK [, BYTE];
: 1006     1923 2
: 1007     1924 2 LOCAL
: 1008     1925 2     REMQUE ADDR,                               ! Address for REMQUE instruction
: 1009     1926 2     CCB_ADDR : REF_BLOCK [0, BYTE],
: 1010     1927 2     BUDDY_CCB : REF_BLOCK [0, BYTE],
: 1011     1928 2     LUN;
: 1012     1929 2
: 1013     1930 2 +
: 1014     1931 2     We now deallocate the LUB/ISB/RAB. An AST will not deallocate under
: 1015     1932 2     us because it will find OTSS$V_IOINPROG set at PUSH time, and will
: 1016     1933 2     therefore set ISBS$V_RECURSIVE so as not to clear OTSS$V_IOINPROG at POP
: 1017     1934 2     time or deallocate the LUB.
: 1018     1935 2 -
: 1019     1936 2     REMQUE_ADDR = OTSS$AA_LUB_TAB [.CCB [LUB$W_LUN], 0];
: 1020     1937 2
: 1021     1938 2     CASE (REMQUE(..REMQUE_ADDR, CCB_ADDR)) FROM 0 TO 3 OF
: 1022     1939 2         SET
: 1023     1940 2
: 1024     1941 2         [0, 3] :
: 1025     1942 2 +
: 1026     1943 2     Zero means that there was more than one entry in the queue.
```

: 1027  
: 1028  
: 1029  
: 1030  
: 1031  
: 1032  
: 1033  
: 1034  
: 1035  
: 1036  
: 1037  
: 1038  
: 1039  
: 1040  
: 1041  
: 1042  
: 1043  
: 1044  
: 1045  
: 1046  
: 1047  
: 1048  
: 1049  
: 1050  
: 1051  
: 1052  
: 1053  
: 1054  
: 1055  
: 1056  
: 1057  
: 1058  
: 1059  
: 1060  
: 1061  
: 1062  
: 1063  
: 1064  
: 1065  
: 1066  
: 1067  
: 1068  
: 1069  
: 1070  
: 1071  
: 1072  
: 1073  
: 1074  
: 1075  
: 1076  
: 1077  
: 1078  
: 1079  
: 1080  
: 1081  
: 1082  
: 1083

1944 2 | This implies that we have done a CLOSE in an AST which went off after  
1945 2 | the INSQUE but before the compensating REMQUE in PUSH\_CCB.  
1946 2 | This should never happen because ISBSV\_RECURSIVE  
1947 2 | will be set in this case.  
1948 2 |  
1949 2 | Three implies that there is nothing in the queue.  
1950 2 | This means that an AST deallocated the LUB, which should not happen  
1951 2 | because of the ISBSV\_RECURSIVE test.  
1952 2 |  
1953 2 | LIB\$STOP (OTSS\_FATINTERR);  
1954 2 |  
1955 2 | [2] :  
1956 2 |  
1957 2 | + The queue is now empty. This is correct. We can now free the LUB.  
1958 2 | Note that PUSH\_CCB will allocate a new LUB if an AST goes off to it  
1959 2 | here, and will carefully not push the LUB we are deallocating.  
1960 2 | First perform a consistency check.  
1961 2 |  
1962 2 |  
1963 2 | IF (.CCB\_ADDR + (.CCB\_ADDR - CCB\_ADDR [LUB\$Q\_QUEUE]) NEQA .CCB) THEN LIB\$STOP (OTSS\_FATINTERR);  
1964 2 |  
1965 2 | [INRANGE, OUTRANGE] :  
1966 2 |  
1967 2 | + This should never happen. The only possible values of REMQUE are  
1968 2 | 0, 2 and 3.  
1969 2 |  
1970 2 | LIB\$STOP (OTSS\_FATINTERR);  
1971 2 | TES:  
1972 2 |  
1973 2 |  
1974 2 | + Since the LUB/ISB/RAB can no longer be used, clear its OTSS\$V\_IOINPROG  
1975 2 | bit. An AST after this point will not indicate recursive I/O.  
1976 2 |  
1977 2 |  
1978 2 | IF (TESTBITCC (OTSS\$V\_IOINPROG [.CCB [LUB\$W\_LUN] - LUB\$K\_ILUN\_MIN])) THEN LIB\$STOP (OTSS\_FATINTERR);  
1979 2 |  
1980 2 |  
1981 2 | + Clear this LUN's buddy's buddy pointer, which points to us.  
1982 2 |  
1983 2 | BUDDY\_CCB = .CCB [LUBSA\_BUDDY\_PTR];  
1984 2 |  
1985 2 | IF (.BUDDY\_CCB NEQA 0) THEN BUDDY\_CCB [LUBSA\_BUDDY\_PTR] = 0;  
1986 2 |  
1987 2 |  
1988 2 | + Free the record buffer if we allocated it.  
1989 2 |  
1990 2 |  
1991 3 | IF (( NOT .CCB [LUB\$V\_USER\_RBUF]) AND (.CCB [LUBSA\_UBF] NEQA 0))  
1992 2 | THEN  
1993 3 | BEGIN  
1994 3 | LOCAL  
1995 3 | FREE\_VM\_STATUS;  
1996 3 |  
1997 3 | FREE\_VM\_STATUS = LIB\$FREE\_VM (%REF (.CCB [LUB\$W\_RBUF\_SIZE]), CCB [LUBSA\_UBF]);  
1998 3 |  
1999 3 | IF ( NOT .FREE\_VM\_STATUS) THEN LIB\$STOP (OTSS\_FATINTERR);  
2000 3 |

```
: 1084      2001 3
: 1085      2002 2      END;
: 1086      2003 2
: 1087      2004 2      !+ Free the file name string, if it is allocated.
: 1088      2005 2      !-
: 1089      2006 2      IF (.CCB [LUB$V_VIRT_RSN])
: 1090      2007 2      THEN
: 1091      2008 3      BEGIN
: 1092      2009 2      LOCAL
: 1093      2010 3      FREE_VM_STATUS;
: 1094      2011 3
: 1095      2012 3
: 1096      2013 3      FREE_VM_STATUS = LIB$FREE_VM (%REF (.CCB [LUB$B_RSL]), CCB [LUB$A_RSN]);
: 1097      2014 3
: 1098      2015 3
: 1099      2016 3      IF ( NOT .FREE_VM_STATUS) THEN LIB$STOP (OTSS_FATINTERR);
: 1100      2017 3
: 1101      2018 3      CCB [LUB$V_VIRT_RSN] = 0;
: 1102      2019 3
: 1103      2020 2      END;
: 1104      2021 2
: 1105      2022 2      !+ Free the prompt buffer, if there is one.
: 1106      2023 2      !-
: 1107      2024 2      BEGIN
: 1108      2025 3
: 1109      2026 3
: 1110      2027 3      LOCAL
: 1111      2028 3      FREE_VM_STATUS;
: 1112      2029 3
: 1113      2030 4      IF ((.CCB [RAB$L_PBF] NEQA 0) AND (.CCB [RAB$V_PMT]))
: 1114      2031 3      THEN
: 1115      2032 4      BEGIN
: 1116      2033 4      FREE_VM_STATUS = LIB$FREE_VM (%REF (LUB$K_PBUF_SIZ), CCB [RAB$L_PBF]);
: 1117      2034 4
: 1118      2035 4      IF ( NOT .FREE_VM_STATUS) THEN LIB$STOP (OTSS_FATINTERR);
: 1119      2036 4
: 1120      2037 3      END;
: 1121      2038 3
: 1122      2039 2      END;
: 1123      2040 2      !+ Remember the logical unit number, since we will need it in a minute.
: 1124      2041 2      !-
: 1125      2042 2      LUN = .CCB [LUB$W_LUN];
: 1126      2043 2
: 1127      2044 2      !+ Now, at last, we can free the CCB itself.
: 1128      2045 2      !-
: 1129      2046 2      BEGIN
: 1130      2047 3
: 1131      2048 3
: 1132      2049 3      LOCAL
: 1133      2050 3      FREE_VM_STATUS;
: 1134      2051 3
: 1135      2052 3      FREE_VM_STATUS = LIB$FREE_VM (%REF (K_TOTAL_CCB_LEN), %REF (.CCB - ISB$K_ISB_LEN - LUB$K_LUB_LEN));
: 1136      2053 3
: 1137      2054 3      IF ( NOT .FREE_VM_STATUS) THEN LIB$STOP (OTSS_FATINTERR);
: 1138      2055 3
: 1139      2056 2      END;
: 1140      2057 2      !+
```

```

: 1141 2058 2 | Since the CCB points to deallocated storage, clear register CCB so
: 1142 2059 2 | that, if anybody refers to it, we will get an access violation.
: 1143 2060 2 |
: 1144 2061 2 | CCB = 0;
: 1145 2062 2 |
: 1146 2063 2 | If the user's program is still running (i.e., if we are not in the
: 1147 2064 2 | exit handler) then the user must have done an explicit CLOSE to cause
: 1148 2065 2 | this LUB to be deallocated. In that case we must clear the LUN
: 1149 2066 2 | allocation so he can do another OPEN on this same logical unit.
: 1150 2067 2 | Note that LUNs less than zero do not have allocation bits since they
: 1151 2068 2 | cannot be opened explicitly by the user.
: 1152 2069 2 |
: 1153 2070 2 |
: 1154 2071 3 IF (.LUN GEQ 0)
: 1155 2072 2 THEN
: 1156 2073 2
: 1157 2074 2 IF ( NOT OTSS$FREE_LUN (LUN) ) THEN LIB$STOP (OTSS_FATINTERR);
: 1158 2075 2
: 1159 2076 2 RETURN;
: 1160 2077 1 END;
                                         ! of routine DEALLOCATE

```

## 003C 00000 DEALLOCATE:

				.WORD	Save R2,R3,R4,R5	1887
				MOVAB	LIB\$FREE_VM, R5	
				MOVAB	LIB\$STOP, R4	
				MOVL	#OTSS_FATINTERR, R3	
				SUBL2	#12, SP	
				CVTWL	-58(CCB), R0	
				MOVAQ	OTSS\$AA_LUB_TAB+64[RO], REMQUE_ADDR	1936
				REMQUE	@0(REMQUE_ADDR), CCB_ADDR	1938
				MOVPSL	RO	
				EXTZV	#1, #2, R0, R0	
				CASEL	RO, #0, #3	
				.WORD	3\$-1\$,-	
					3\$-1\$,-	
					2\$-1\$,-	
					3\$-1\$	
50	50	02	01	OE	11 0003D	1953
0018	000A	0018	50	52	C3 0003F 2\$: BRB	1963
				52	9E 00043	
				58	A042	
				58	D1 00048	
					L 13 0004B	
					50 DD 0004D 3\$: BEQL	
				64	01 FB 0004F	
				50	AB 32 00052 4\$: CALLS	
				50	08 C0 00056	
				05 0000000G	50 E4 00059	
				64	DD 00061	
				50	01 FB 00063	
				B8	AB DD 00066 5\$: CALLS	
				50	03 13 0006A	
				B8	A0 D4 0006C	
				FF	AB 95 0006F 6\$: CLRL	
					TSTA -1(CCB)	

			18	19	00072	BLSS	7\$		
			9C	AB	D5 00074	TSTL	-100(CC8)		
				16	13 00077	BEQL	7\$		
			9C	AB	9F 00079	PUSHAB	-100(CC8)	1998	
			D2	AB	3C 0007C	MOVZWL	-46(CC8), 8(SP)		
			08	AE	9F 00081	PUSHAB	8(SP)		
			65	02	FB 00084	CALLS	#2 LIB\$FREE_VM		
			05	50	E8 00087	BLBS	FREE_VM_STATUS, 7\$	2000	
				53	DD 0008A	PUSHL	R3		
			64	01	FB 0008C	CALLS	#1, LIB\$STOP		
			1A	FE	AB E9 0008F	BLBC	-2(CC8), 9\$	2008	
				F8	AB 9F 00093	PUSHAB	-8(CC8)	2015	
			08	AE	9A 00096	MOVZBL	-9(CC8), 8(SP)		
			65	02	FB 0009E	CALLS	#2 LIB\$FREE_VM		
			05	50	E8 000A1	BLBS	FREE_VM_STATUS, 8\$	2017	
				53	DD 000A4	PUSHL	R3		
			64	01	FB 000A6	CALLS	#1, LIB\$STOP		
			FE	AB	8A 000A9	BICB2	#1, -2(CC8)	2019	
				30	AB D5 000AD	TSTL	48(CC8)	2030	
					1B 13 000B0	BEQL	10\$		
			16	07	AB 06 E1 000B2	BBC	#6, 7(CC8), 10\$		
				08	AE 30 AB 9F 000B7	PUSHAB	48(CC8)	2033	
					50 8F 9A 000BA	MOVZBL	#80, 8(SP)		
				65	02 FB 000C2	PUSHAB	8(SP)		
				05	50 E8 000C5	CALLS	#2 LIB\$FREE_VM		
					53 DD 000C8	BLBS	FREE_VM_STATUS, 10\$	2035	
				64	01 FB 000CA	PUSHL	R3		
				08	AE AB 32 000CD	CALLS	#1, LIB\$STOP		
				04	AE FEE0 CB 9E 000D2	CVTWL	-58(CC8), LUN	2043	
					04 AE 9F 000D8	MOVAB	-288(R11), 4(SP)	2052	
				04	0164 8F 3C 000DB	PUSHAB	4(SP)		
					04 AE 9F 000E1	MOVZWL	#356, 4(SP)		
				65	02 FB 000E4	PUSHAB	4(SP)		
				05	50 E8 000E7	CALLS	#2 LIB\$FREE_VM		
					53 DD 000EA	BLBS	FREE_VM_STATUS, 11\$	2054	
				64	01 FB 000EC	PUSHL	R3		
					58 D4 000EF	CALLS	#1, LIB\$STOP		
				08	AE D5 000F1	CLRL	CCB	2061	
					12 19 000F4	TSTL	LUN	2071	
				08	AE 9F 000F6	BLSS	12\$		
			000000006	00	01 FB 000F9	PUSHAB	LUN	2074	
				05	50 E8 00100	CALLS	#1, OTSS\$FREE_LUN		
					53 DD 00103	BLBS	R0, 12\$		
				64	01 FB 00105	PUSHL	R3		
					04 00108	CALLS	#1, LIB\$STOP		
						RET		2077	

: Routine Size: 265 bytes, Routine Base: \_OTSS\$CODE + 02F6

```
: 1162      2078 1 ROUTINE POP_ACTIVE : CALL_CCB NOVALUE =           ! Pop old active unit
: 1163      2079 1
: 1164      2080 1 ++
: 1165      2081 1   FUNCTIONAL DESCRIPTION:
: 1166      2082 1
: 1167      2083 1     Restore the status of an interrupted I/O statement using the
: 1168      2084 1     information saved when the statement was interrupted. All of
: 1169      2085 1     the ISB is restored, and a few other things. In some unusual
: 1170      2086 1     cases there is no CCB to restore to, so only OTSSL_CUR_LUN is
: 1171      2087 1     restored.
: 1172      2088 1
: 1173      2089 1   CALLING SEQUENCE:
: 1174      2090 1
: 1175      2091 1     CALL POP_ACTIVE ()
: 1176      2092 1
: 1177      2093 1   FORMAL PARAMETERS:
: 1178      2094 1
: 1179      2095 1     NONE
: 1180      2096 1
: 1181      2097 1   IMPLICIT INPUTS:
: 1182      2098 1
: 1183      2099 1     The DEALLOC bit in the LUB
: 1184      2100 1
: 1185      2101 1   IMPLICIT OUTPUTS:
: 1186      2102 1
: 1187      2103 1     The ISB, and some other fields of the CCB
: 1188      2104 1     CCB          The restored CCB
: 1189      2105 1
: 1190      2106 1   SIDE EFFECTS:
: 1191      2107 1
: 1192      2108 1     Calls LIB$FREE_VM to free virtual memory.
: 1193      2109 1     ---
: 1194      2110 1
: 1195      2111 2   BEGIN
: 1196      2112 2
: 1197      2113 2   EXTERNAL REGISTER
: 1198      2114 2     CCB : REF_BLOCK [, BYTE];
: 1199      2115 2
: 1200      2116 2   LOCAL
: 1201      2117 2     PUSH : REF_BLOCK [PUSH$K_LENGTH, BYTE] FIELD (PUSH_ITEM),
: 1202      2118 2           ! Logical unit number being restored
: 1203      2119 2
: 1204      2120 2     +
: 1205      2121 2     Get an activation record off the I/O Active queue. It had better
: 1206      2122 2     be there.
: 1207      2123 2     -
: 1208      2124 2
: 1209      2125 2     IF (REMQUE (.OTSSQ_IO_ACTIVE [0], PUSH)) THEN LIB$STOP (OTSS_FATINTERR);
: 1210      2126 2
: 1211      2127 2     +
: 1212      2128 2     Fetch the logical unit number associated with this record.
: 1213      2129 2     -
: 1214      2130 2     LUN = .PUSH [PUSH$W_LUN];
: 1215      2131 2     +
: 1216      2132 2     If this is a fake activation record, just store the LUN.
: 1217      2133 2     -
: 1218      2134 2
```

```
: 1219      2135 3     IF (.PUSH [PUSHSV_FAKE])
: 1220      2136 3     THEN
: 1221      2137 3     OTSSSL_CUR_LUN = .LUN
: 1222      2138 3
: 1223      2139 3     ELSE
: 1224      2140 3     BEGIN
: 1225      2141 3     |+ If this LUN does not have I/O in progress then something is very
: 1226      2142 3     |+ wrong.
: 1227      2143 3
: 1228      2144 3
: 1229      2145 3     IF ( NOT .OTSSSV_I0INPROG [.LUN - LUB$K_ILUN_MIN] ) THEN LIB$STOP (OTSS_FATINTERR);
: 1230      2146 3
: 1231      2147 3     |+
: 1232      2148 3     |+ There was previous I/O. Restore the ISB, etc of the pushed unit.
: 1233      2149 3     |+ Because of ASTs, we must store OTSSSL_CUR_LUN before copying
: 1234      2150 3     |+ data from the I/O Active entry because only the LUN indicated by
: 1235      2151 3     |+ OTSSSL_CUR_LUN will get pushed.
: 1236      2152 3
: 1237      2153 3     |-
: 1238      2154 3     |+ OTSSSL_CUR_LUN = .LUN;
: 1239      2155 3     |+ CCB = .OTSSAA_LUB_TAB [.LUN, 0];
: 1240      2156 3     |+ CCB = .CCB [0, 0, 0, 0] - CCB [LUBSQ_QUEUE];
: 1241      2157 3     |+ OTSSSA_CUR_LUB = .CCB;
: 1242      2158 3     |+ CCB [L0BSV_I0_ACTIVE] = .PUSH [PUSH$V_I0_ACT];
: 1243      2159 3     |+ CCB [RABSL_STS] = .PUSH [PUSH$L_STS];
: 1244      2160 3     |+ CCB [RABSL_STV] = .PUSH [PUSH$L_STV];
: 1245      2161 3     |+ CCB [RAB$B_TMO] = .PUSH [PUSH$B_TMO];
: 1246      2162 4
: 1247      2163 3     IF (.PUSH [PUSH$V_PMT])
: 1248      2164 4     THEN
: 1249      2165 4     BEGIN
: 1250      2166 4     |+ CCB [RAB$B_PSZ] = .PUSH [PUSH$B_PSZ];
: 1251      2167 4     |+ CH$MOVE (.CCB [RAB$B_PSZ], PUSH[PUSH$T_PROMPT], .CCB [RABSL_PBF]);
: 1252      2168 3
: 1253      2169 3     |+
: 1254      2170 3     |+ CH$MOVE (ISBSK_ISB_LEN, PUSH [PUSH$X_ISB], .CCB - ISBSK_ISB_LEN - LUB$K_LUB_LEN);
: 1255      2171 3     |+
: 1256      2172 3     |+ If the LUN has been marked for deallocation (which means that it
: 1257      2173 3     |+ has been closed but not deallocated yet because it has I/O in
: 1258      2174 3     |+ progress) then clear the statement type field so that all
: 1259      2175 3     |+ continued I/O will fail. The statement type must be set so that
: 1260      2176 3     |+ the owning language will get an error when I/O continues.
: 1261      2177 3
: 1262      2178 4
: 1263      2179 3     IF (.CCB [LUB$V DEALLOC])
: 1264      2180 3     THEN
: 1265      2181 3     CASE .CCB [LUB$B_LANGUAGE] FROM LUB$K_LANG_MIN TO LUB$K_LANG_MAX OF
: 1266      2182 3     SET
: 1267      2183 3
: 1268      2184 3     |+ [LUB$K_LANG_FOR]:
: 1269      2185 3     |+ CCB [ISBSB_STM_TYPE] = ISBSK_FORSTTYLO - 1;
: 1270      2186 3
: 1271      2187 3     |+ [LUB$K_LANG_BAS]:
: 1272      2188 3     |+ CCB [ISBSB_STM_TYPE] = ISBSK_BASSTTYLO - 1;
: 1273      2189 3
: 1274      2190 3     |+ [LUB$K_LANG_NONE]:
: 1275      2191 3     |+ CCB [ISBSB_STM_TYPE] = 0;
```

```

: 1276 2192 3
: 1277 2193 3
: 1278 2194 3
: 1279 2195 3
: 1280 2196 3
: 1281 2197 2
: 1282 2198 2
: 1283 2199 2
: 1284 2200 2 + [OUTRANGE] :
: 1285 2201 2 - LIB$STOP (OTSS_FATINTERR);
: 1286 2202 3 TES;
: 1287 2203 3
: 1288 2204 3
: 1289 2205 3 END;
: 1290 2206 3
: 1291 2207 3
: 1292 2208 3
: 1293 2209 3
: 1294 2210 3
: 1295 2211 2
: 1296 2212 2
: 1297 2213 1 RETURN: ! of routine POP_ACTIVE
    END;

```

03FC 00000 POP_ACTIVE						
				WORD	Save R2,R3,R4,R5,R6,R7,R8,R9	2078
		59 00000000G	00 9E 00002	MOVAB	OTSSSL_CUR_LUN, R9	
		58 00C00000G	00 9E 00009	MOVAB	LIB\$STOP, R8	
		57 00000000G	8F D0 00010	MOVL	#OTSS_FATINTERR, R7	
		5E	08 C2 00017	SUBL2	#8, SP	
	04	AE 00000000'	FF 0F 0001A	REMQUE	#OTSSQ_IO_ACTIVE, PUSH	2125
			05 1C 00022	BVC	1\$	
			57 DD 00024	PUSHL	R7	
		68	01 FB 00026	CALLS	#1, LIB\$STOP	
		56 14	04 AE D0 00029	1\$:	MOVL	2130
		A6	01 A6 32 0002D	CVTWL	PUSH, R6	
		52	10 A6 32 00031	BBC	16(R6), LUN	2135
		69	52 D0 00036	MOVL	#1, 20(R6), 2\$	
			7E 11 00039	BRB	LUN, OTSSSL_CUR_LUN	2137
	05	53 00000000G	08 A2 9E 0003B	MOVAB	8(R2), R3	2145
		00	53 E0 0003F	BBS	R3, OTSSSV_IOINPROG, 3\$	
			57 DD 00047	PUSHL	R7	
		68	01 FB 00049	CALLS	#1, LIB\$STOP	
		69	52 D0 0004C	3\$:	MOVL	2153
		00000000G0043	7F 0004F	PUSHAQ	LUN, OTSSSL_CUR_LUN	
		5B	9E D0 00056	MOVL	OTSSSA_LUB_TAB[R3]	2154
FC AB	50	5B	5B C3 00059	SUBL3	a(SP)+, CCB	
		5B	58 A04B 9E 0005D	MOVAB	CCB, CCB, R0	2155
	01	00000000G	00 5B D0 00062	MOVL	88(R0)[CCB], CCB	
		01	14 A6 F0 00069	INSV	CCB, OTSSSA_CUR_LUB	2156
		08 AB	08 A6 7D 00070	MOVQ	20(R6), #1, #1, -4(CCB)	2157
		1F AB	13 A6 90 00075	MOVB	8(R6), 8(CCB)	2158
	0F	14 A6	02 E1 0007A	BBC	19(R6), 31(CCB)	2160
		34 AB	12 A6 90 0007F	MOVB	#2, 20(R6), 4\$	2162
					18(R6), 52(CCB)	2165

30	BB	15	50	34	AB	9A	00084	MOVZBL	52(CC(B)) R0	: 2166	
FEC0	CB	65	A6	008C	50	28	00088	MOVC3	R0 21(R6) @48(CC(B))		
1D		FF	A6	008E	8F	28	0008E	MOVC3	#188, 101(R6) -298(CC(B))		
02			AB	48	E1	00097	48:	BBC	#4 -1(CC(B)) 88		
0014		000D	00	D8	AB	8F	0009C	CASEB	-40(CC(B)), #0, #2		
				0014		000A1	58:	.WORD	7\$-5\$,- 6\$-5\$,- 7\$-5\$		
					57	DD	000A7	PUSHL	R7	2194	
					01	FB	000A9	CALLS	#1, LIB\$STOP		
					0B	11	000AC	BRB	8\$		
		FF71	CB		1A	90	000AE	68:	MOVB	#26, -143(CC(B))	
					04	11	000B3	BRB	8\$		
				FF71	CB	94	000B5	78:	CLRB	-143(CC(B))	
					04	AE	000B9	88:	PUSHAB	PUSH	
			04	0121	8F	3C	000BC	MOVZWL	#289, 4(SP)		
			04		AE	9F	000C2	PUSHAB	4(SP)		
		00000000G	00		02	FB	000C5	CALLS	#2, LIB\$FREE_VM		
			05		50	E8	000CC	BLBS	FREE_VM_RESULT, 9\$		
					57	DD	000CF	PUSHL	R7	2209	
					01	FB	000D1	CALLS	#1, LIB\$STOP		
					04	000D4	98:	RET		2213	

: Routine Size: 213 bytes, Routine Base: \_OTSSCODE + 03FF

: 1298 2214 1

: 1300 2215 1 GLOBAL ROUTINE OTSS\$POP\_CCB ! Restore old CCB  
: 1301 2216 1 : JSB\_CB\_POP NOVALUE =  
: 1302 2217 1 ++  
: 1303 2218 1 FUNCTIONAL DESCRIPTION:  
: 1304 2219 1  
: 1305 2220 1  
: 1306 2221 1 Restore the I/O system to its state before the call to  
: 1307 2222 1 PUSH\_CCB. Clear LUB\$V\_IO\_ACTIVE. If the I/O active list  
: 1308 2223 1 is empty, clear OTSSA\_CUR\_LUB, otherwise set it to  
: 1309 2224 1 its previous value and restore its ISB, etc.  
: 1310 2225 1  
: 1311 2226 1 If virtual memory for a compiled format is allocated for this  
: 1312 2227 1 ISB, it is freed.  
: 1313 2228 1  
: 1314 2229 1 CALLING SEQUENCE:  
: 1315 2230 1 CALL OTSS\$POP\_CCB ()  
: 1316 2231 1  
: 1317 2232 1 FORMAL PARAMETERS:  
: 1318 2233 1  
: 1319 2234 1  
: 1320 2235 1  
: 1321 2236 1  
: 1322 2237 1  
: 1323 2238 1  
: 1324 2239 1  
: 1325 2240 1 CCB  
: 1326 2241 1 OTSSAA\_LUB\_TAB  
: 1327 2242 1 OTSSQ\_IO\_ACTIVE  
: 1328 2243 1  
: 1329 2244 1 IMPLICIT OUTPUTS:  
: 1330 2245 1 CCB Set to previous LUB/ISB/RAB  
: 1331 2246 1 OTSSQ\_IO\_ACTIVE Holds one fewer item  
: 1332 2247 1 LUB\$V\_IO\_ACTIVE Cleared to indicate I/O no longer active,  
: 1333 2248 1 but may be set by the pop from the  
: 1334 2249 1 I/O Active list.  
: 1335 2250 1  
: 1336 2251 1 SIDE EFFECTS:  
: 1337 2252 1  
: 1338 2253 1 May call LIB\$FREE\_VM to free virtual memory.  
: 1339 2254 1 --  
: 1340 2255 1  
: 1341 2256 2 BEGIN  
: 1342 2257 2  
: 1343 2258 2 EXTERNAL REGISTER  
: 1344 2259 2 CCB : REF BLOCK [, BYTE];  
: 1345 2260 2  
: 1346 2261 2 ++  
: 1347 2262 2 If the LUB has been marked for deallocation (by CLOSE) and there is  
: 1348 2263 2 no I/O active, deallocate it. If there is I/O Active, the  
: 1349 2264 2 deallocation must be defered until after all of the I/O has completed  
: 1350 2265 2 to insure that the continued I/O will get the "I/O continued to closed  
: 1351 2266 2 file" error.  
: 1352 2267 2 --  
: 1353 2268 2  
: 1354 2269 3 IF (.CCB [LUB\$V DEALLOC] AND ( NOT .CCB [ISBV RECURSIVE]))  
: 1355 2270 2 THEN  
: 1356 2271 2 DEALLOCATE ()

```
: 1357      2272 2      ELSE
: 1358      2273 2      BEGIN
: 1359      2274 3      + This is no longer the unit with I/O active.
: 1360      2275 3      -
: 1361      2276 3      CCB [LUB$V_IO_ACTIVE] = 0;
: 1362      2277 3      +
: 1363      2278 3      See if I/O will continue on this unit. It will continue if
: 1364      2279 3      ISBSV_RECURSIVE is set, which means that PUSH_CCB was called
: 1365      2280 3      with I/O in progress on this LUN. We make this test before
: 1366      2281 3      restoring the ISB because we may be restoring to the same
: 1367      2282 3      LUN, and the former I/O may be the top level of I/O for this
: 1368      2283 3      LUN, and if so it will have ISBSV_RECURSIVE clear.
: 1369      2284 3      -
: 1370      2285 3      -
: 1371      2286 3      IF (.OTSS$L_LVL_CTR EQ 0) AND (.CCB [ISBSV_RECURSIVE]) THEN LIB$STOP (OTSS_FATINTERR);
: 1372      2287 3      IF ( NOT .CCB [ISBSV_RECURSIVE])
: 1373      2288 3      THEN
: 1374      2289 4      IF (TESTBITCC (OTSS$V_IOINPROG [.CCB [LUB$W_LUN] - LUB$K_ILUN_MIN]))
: 1375      2290 3      THEN
: 1376      2291 3      LIB$STOP (OTSS_FATINTERR);
: 1377      2292 4      LIB$STOP (OTSS_FATINTERR);
: 1378      2293 3      -
: 1379      2294 3      -
: 1380      2295 3      -
: 1381      2296 2      END;
: 1382      2297 2      -
: 1383      2298 2      +
: 1384      2299 2      Since OTSS$V_IOINPROG may now be clear, our CCB may be deallocated, so
: 1385      2300 2      we cannot touch it again. For that matter, we may have deallocated
: 1386      2301 2      it ourselves above.
: 1387      2302 2      If there was previous I/O, restore it. Otherwise return to the idle
: 1388      2303 2      state.
: 1389      2304 2      -
: 1390      2305 2      -
: 1391      2306 2      -
: 1392      2307 3      IF (.OTSS$L_LVL_CTR NEQ 0)
: 1393      2308 2      THEN
: 1394      2309 2      POP_ACTIVE ()
: 1395      2310 2      ELSE
: 1396      2311 3      BEGIN
: 1397      2312 3      OTSS$A_CUR_LUB = 0;
: 1398      2313 3      OTSS$L_CUR_LUN = LUB$K_LUN_MAX + 1;
: 1399      2314 2      END;
: 1400      2315 2      -
: 1401      2316 2      +
: 1402      2317 2      Decrement the level counter. If we are at the top level the level
: 1403      2318 2      counter will go from 0 to -1.
: 1404      2319 2      -
: 1405      2320 2      OTSS$L_LVL_CTR = .OTSS$L_LVL_CTR - 1;
: 1406      2321 2      RETURN;
: 1407      2322 1      END;                                ! of routine OTSS$POP_CCB
```

FE14	07	97	AB	E8 00005	BBC	#4 -1(CCB), 1\$	: 2269
	CF		00	FB 00009	BLBS	-105(CCB), 1\$	
			3D	11 0000E	CALLS	#0, DEALLOCATE	: 2271
FC	AB	02	8A 00010	1\$: 11	BRB	4\$	: 2277
		00000000G	00	D5 00014	TSTL	OTSS\$L_LVL_CTR	: 2287
			11	12 0001A	BNEQ	2\$	
		00000000G	11	12 0001C	BLBC	-105(CCB), 3\$	
			00000000G	8F DD 00020	PUSHL	#OTSS\$ FATINTERR	
00000000G	00	01	FB 00026	CALLS	#1 LIBSTOP		
	1C	97	AB E8 0002D	2\$: 50	BLBS	-105(CCB), 4\$	: 2289
		C6	AB 52 00031	3\$: 50	CVTWL	-58(CCB), R0	: 2292
			08 CO 00035	ADDL2	#8, R0		
0D	00000000G	00	50 E4 00038	BBSC	R0, OTSS\$V IOINPROG, 4\$		
	00000000G	00	8F DD 00040	PUSHL	#OTSS\$ FATINTERR		: 2294
	00000000G	00	01 FB 00046	CALLS	#1 LIBSTOP		
		00000000G	00 D5 0004D	4\$: 07	TSTL	OTSS\$L_LVL_CTR	: 2307
FED1	CF	00	FB 00055	BEQL	5\$		: 2309
			OE 11 0005A	BRB	6\$		
	00000000G	00	D4 0005C	5\$: 78	CLRL	OTSS\$A CUR_LUB	: 2312
	00000000G	00	8F 9A 00062	6\$: 0000000G	MOVZBL	#120, OTSS\$L CUR_LUN	: 2313
			D7 0006A	6\$: 05 00070	DECL	OTSS\$L_LVL_CTR	: 2320
					RSB		: 2322

; Routine Size: 113 bytes, Routine Base: \_OTSS\$CODE + 04D4

```
: 1408 2323 1
: 1409 2324 1 END
: 1410 2325 1
: 1411 2326 0 ELUDOM
```

!End of module OTSS\$CCB

## PSECT SUMMARY

Name	Bytes	Attributes
_OTSS\$DATA	8	NOVEC, WRT, RD .NOEXE,NOSHR, LCL, REL, CON, PIC,ALIGN(2)
_OTSS\$CODE	1349	NC'EC,NOWRT, RD : EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

## Library Statistics

File	----- Symbols -----	Pages Mapped	Processing Time
	Total      Loaded      Percent		
\$_255\$DUA28:[SYSLIB]STARLET.L32;1	9776      20      0	581	00:00.8

## COMMAND QUALIFIERS

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

: Size: 1349 code + 8 data bytes  
: Run Time: 00:21.3  
: Elapsed Time: 01:31.4  
: Lines/CPU Min: 6567  
: Lexemes/CPU-Min: 37228  
: Memory Used: 185 pages  
: Compilation Complete

0211 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

OTSCUB  
LIS

OTSCCBOAT  
LIS

OTSCUTDP  
LIS

OTSCUTFP  
LIS

OTSCUTLT  
LIS

LIBVECTR2  
LIS

OTSCNVOUT  
LIS

OTSCVHP  
LIS

LIBWAIT  
LIS

OTSCLOSEF  
LIS

OTSCUTDT  
LIS

LIBVECTOR  
LIS

LIBUM  
LIS

OTSCUTGP  
LIS