D 8

```
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III        BBB         BBB  RRR      RRR           TTT        LLL
LLL                    III        BBB         BBB  RRR      RRR           TTT        LLL
LLL                    III        BBB         BBB  RRR      RRR           TTT        LLL
LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

B 6

FILEID**OTSCCB

```
  000000    TTTTTTTTTT    SSSSSSSS    CCCCCCCC    CCCCCCCC  BBBBBBBB
  000000    TTTTTTTTTT    SSSSSSSS    CCCCCCCC    CCCCCCCC  BBBBBBBB
00      00      TT      SS          CC          CC          BB      BB
00      00      TT      SS          CC          CC          BB      BB
00      00      TT      SS          CC          CC          BB      BB
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
00      00      TT              SS  CC          CC          BB      BB
00      00      TT              SS  CC          CC          BB      BB      ....
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      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
```

```
0 MODULE OTS$$CCB (                                      ! Push, Pop, Allocate, and deallocate LUB/ISB/RAB
0                 IDENT = '1-057'                        ! File: OTSCCB.B32 Edit: FM1057
0                 ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 ! FACILITY: language support library
1 !
1 ! ABSTRACT:
1 !
1 !       This module supports pushing and popping of the CCB, the
1 !       common control block for the I/O part of the RTL.  Currently,
1 !       only BASIC uses this module, since FORTRAN does its own
1 !       manipulations.
1 !
1 ! ENVIRONMENT: User mode, AST level or not or mixed
1 !
1 ! AUTHOR:  Thomas N. Hastings, CREATION DATE: 01-June-77
1 !
1 ! MODIFIED BY:
1 !
1 !       Thomas N. Hastings, 01-June-77: VERSION 01
1 ! 01    -  original
1 ! 0-26  - Set RMS RAB$V_UIF bit TNH 19-SEP-77
1 ! 0-27  - Set RMS RAB$V_TPT bit (truncate on sequential $PUT not at EOF TNH 24-SEP-77
1 ! 0-28  - Use FOR$$SIG_NO_LUB since no LUB.  TNH 24-SEP-77
1 ! 0-30  - Set RAB bits for read-ahead, write-behind, locate mode JMT 21-OCT-77
1 ! 0-31  - Use FOR$K_abcmnoxyz as EXTERNAL LITERALs.  TNH 27-Oct-77
1 ! 0-32  - Made second arg optional.  TNH 9-Nov-77
1 ! 0-33  - Use OTS$_FATINTERR.  TNH 01-Dec-77
1 ! 0-34  - Clear FAB after call to LIB$GET_VM.  TNH 9-Dec-77
1 ! 0-35  - Call FOR$$SIG_FATINT.  TNH 30-Dec-77
1 ! 0-36  - Have CB_POP signal FATINT if LUB not active;
1 !           Add routine CB_CND_POP to conditionally pop if LUB active,
1 !             otherwise NO-OP (OTS exit handler calls this).  JMT 10-Jan-78
1 ! 0-37  - Remove CB_CND_POP; I didn't really want it, anyway...  JMT 11-Jan-78
1 ! 0-37  - Global register CCB.  JMT 8-Apr-78
1 ! 0-39  - Change to STARLET library.  DGP 20-Apr-78
1 ! 0-40  - Change REQUIRE files for VAX system build.  DGP 28-Apr-78
1 ! 0-41  - Change STARLET to RTLSTARLE to avoid conflicts.  DGP 1-May-78
1 ! 0-42  - Make JSB linkage.  TNH 19-May-78
1 ! 0-46  - Use FOR$$GET_VM with new optional 2nd arg.  TNH 21-May-78
1 ! 0-47  - Remove setting ISB to -1.  TNH 30-May-78.
1 ! 0-48  - Add sanity check of data base.  TNH 10-June-78
1 ! 0-49  - Add call to FOR$$SIG_DATCOR.  TNH 10-June-78
1 ! 0-50  - Add FOR$$CB_GET entry for non-shared access to OTS$$A_CUR_LUB. TNH 2-Aug-78
1 ! 0-52  - Fix AST re-entrant timing hole.  TNH 9-Aug-78
1 ! 0-53  - Change file name to FORCB.B32, and change the names of the
1 !           REQUIRE files similarly.  JBS 14-NOV-78
1 ! 1-001 - Update version number and copyright notice.  JBS 16-NOV-78
1 ! 1-002 - Change LUB$B_LUN to LUB$W_LUN.  JBS 05-DEC-78
1 ! 1-003 - Change REQUIRE file names from FOR... to OTS...  JBS 07-DEC-78
1 ! 1-004 - Include TNH's version, which uses a bit table to provide
1 !           AST re-entrancy.  JBS 11-DEC-78
1 ! 1-005 - Remove REQUIRE of OTSMAC; not needed.  JBS 11-DEC-78
1 ! 1-006 - Add FOR$$CB_NEXT, which gets the next LUN for the CLOSE loop
1 !           in FOROPEN.B32.  JBS 11-DEC-78
1 ! 1-007 - Fix coding errors in FOR$$CB_NEXT and make OTS$$AA_LUB_TAB
1 !           OWN.  JBS 18-DEC-78
1 ! 1-008 - Change file and module name to OTSCB and add specialized
1 !           BASIC entry points.  This is in preparation for recursive
1 !           I/O.  JBS 29-DEC-78
1 ! 1-009 - Add BAS$$CB_CLEANUP.  JBS 29-DEC-78
1 ! 1-010 - Add recursive I/O for BASIC.  JBS 08-JAN-1979
1 ! 1-011 - Divide into three modules: OTSCCB, FORCB and BASCB.  This
1 !           module, OTSCCB, contains the language-independent code.
1 !           JBS 09-JAN-1979
1 ! 1-012 - Restore OTS$$A_CUR_LUB and set I/O Active when popping.
1 !           JBS 15-JAN-1979
1 ! 1-013 - Fix up some complex cases of popping recursive I/O.
1 !           JBS 15-JAN-1979
1 ! 1-014 - Fix an error in calling LIB$STOP.  JBS 16-JAN-1979
1 ! 1-015 - Push and Pop the RMS timeout field in the RAB.  JBS 16-JAN-1979
1 ! 1-016 - Use the DEALLOC bit in the LUB to interlock deallocation of
1 !           the LUB/ISB/RAB rather than disabling interrupts, and be
1 !           cleverer in other places so that interrupts need never be
1 !           disabled.  JBS 23-JAN-1979
1 ! 1-017 - Don't clear OTS$$V_IOINPROG if the LUN we just popped is the same
1 !           unit we just finished using.  (This is the most common case
1 !           of recursive I/O.)  JBS 24-JAN-1979
1 ! 1-018 - But if there is no popped unit, do clear OTS$$V_IOINPROG.  (This is
1 !           the most common case of non-recursive I/O!).  JBS 24-JAN-1979
1 ! 1-019 - Divide into more internal subroutines in an attempt to speed
1 !           up the common pushing and popping cases by avoiding the
1 !           saving of unnecessary registers.  JBS 25-JAN-1979
1 ! 1-020 - Change linkage for OTS$PUSH_CCB to JSB_CB_PUSH and for
1 !           OTS$POP_CCB to JSB_CB_POP.  JBS 25-JAN-1979
1 ! 1-021 - Clear length of prompt buffer when pushing.  JBS 26-JAN-1979
1 ! 1-022 - Remove OTS$CLEANUP_IO, we will clean I/O using a stack
1 !           frame instead.  JBS 26-JAN-1979
1 ! 1-023 - Change to double dollar signs since these entry points are
```

```
1 !          not for use by users.  JBS 26-JAN-1979
1 ! 1-024 - Deallocate the LUN after the LUB/ISB/RAB has been deallocated.
1 !          Note that OPEN allocates it.  JBS 26-JAN-1979
1 ! 1-025 - Make the table storage PIC, even though it is used by INSQUE
1 !          and REMQUE instructions, by initializing it at run time.
1 !          This requires disabling ASTs during initialization, but it is
1 !          done only once per image activation.  JBS 28-JAN-1979
1 ! 1-026 - Rearrange the order of some of the manipulations to make
1 !          PUSH and POP really AST re-entrant.  JBS 29-JAN-1979
1 ! 1-027 - Make these routines AST reentrant
1 !          in the face of deallocation at AST level.  JBS 31-JAN-1979
1 ! 1-028 - If LUB$V_USER_RBUF is set, don't deallocate the record
1 !          buffer, it belongs to the user!  JBS 16-FEB-1979
1 ! 1-029 - Clear the buddy's buddy pointer, which points to us, when
1 !          deallocating.  JBS 16-FEB-1979
1 ! 1-030 - Print an error message if the ISB overlaps the LUB.  This can
1 !          happen if the LUB is extended but the ISB is not edited to
1 !          reflect it.  JBS 21-MAR-1979
1 ! 1-031 - Initialize LUB$Q_BFA_QUEUE.  JBS 05-APR-1979
1 ! 1-032 - Don't free the file name string unless it has been allocated
1 !          in virtual memory.  JBS 10-APR-1979
1 ! 1-033 - Don't free the record buffer unless it has been allocated.
1 !          JBS 10-APR-1979
1 ! 1-034 - Free the compiled format, if allocated.  SBL 27-Apr-1979
1 ! 1-035 SBL1035 - Set ISB$W_FMT_LEN to zero on allocation.  SBL 4-May-79
1 ! 1-036 - Change CASE off result of REMQUE to match what is
1 !         actually given by that function.  SBL 9-May-1979
1 ! 1-037 - Change required file name to OTSCCBREQ so as not to conflict
1 !         with this module at system build time.  SBL 10-May-1979
1 ! 1-038 - Move clearing of ISB$W_FMT_LEN to allocation stage.  SBL 14-May-1979
1 ! 1-039 - Fix bug in compiled format deallocation.  SBL 15-May-1979
1 ! 1-040 - Fix another one.  Length must be passed as address of a word!
1 ! 1-041 - We overlooked the REMQUE in DEALLOCATE.  SBL 17-May-1979
1 !         So, we have to construct a temp.  SBL 17-May-1979
1 ! 1-042 - Clear ISB$W_FMT_LEN during PUSH, so that POP won't try
1 !          to deallocate the format prematurely.  JBS 29-MAY-1979
1 ! 1-043 - Set up LUB$A_BUDDY_PTR during allocate.  JBS 30-MAY-1979
1 ! 1-044 - Make much of the data structure global so it can be
1 !          referenced directly by FOR$$CB.  JBS 28-JUN-1979
1 ! 1-045 - Do an RMS $WAIT if there is I/O active on the unit we
1 !          are starting.  JBS 25-JUL-1979
1 ! 1-046 - Don't make OTS$Q_IO_ACTIVE global.  JBS 26-JUL-1979
1 ! 1-047 - Save the prompt buffer only if it really is a prompt buffer.
1 !          If it is a key buffer, the key may be in read-only storage.
1 !          JBS 09-AUG-1979
1 ! 1-048 - Move the global parts of the data base to OTS$$CCB_DATA
1 !          so this module need not be loaded if only FORTRAN programs
1 !          are in the image.  JBS 16-AUG-1979
1 ! 1-049 - Return CCB as 0 from POP to indicate deallocation.  JBS 17-AUG-1979
1 ! 1-050 - Correct an error in a comment.  JBS 10-SEP-1979
1 ! 1-051 - When deallocating, LUB$A_BUF_BEG points to the buffer; in locate
1 !          mode, LUB$A_RBUF_ADR may point to RMS space.  JBS 13-SEP-1979
1 ! 1-052 - Remove the references to ISB$W_FMT_LEN; now done in FORCB.
1 !          JBS 18-SEP-1979
1 ! 1-053 - Remove references to LUB$Q_BFA_QUEUE; no longer used.
1 !          JBS 18-SEP-1979
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$$IO_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  !
;  193          0620  1  ! TABLE OF CONTENTS:
;  194          0621  1  !
;  195          0622  1
;  196          0623  1  FORWARD ROUTINE
;  197          0624  1      INITIALIZE : NOVALUE,                         ! Set up the LUB table and the active queue
;  198          0625  1      PUSH_FAKE : CALL_CCB,                         ! Push fake record
;  199          0626  1      PUSH_ACTIVE : CALL_CCB,                       ! Push active LUB
;  200          0627  1      ALLOCATE : CALL_CCB,                          ! Allocate LUB/ISB/RAB
;  201          0628  1      OTS$$PUSH_CCB : JSB_CB_PUSH,                  ! Get the CCB, push old use of it
;  202          0629  1      DEALLOCATE : CALL_CCB NOVALUE,                ! Deallocate LUB/ISB/RAB
;  203          0630  1      POP_ACTIVE : CALL_CCB NOVALUE,                ! Pop active LUN
;  204          0631  1      OTS$$POP_CCB : JSB_CB_POP NOVALUE;            ! Restore old use of CCB
;  205          0632  1
;  206          0633  1  !
;  207          0634  1  ! INCLUDE FILES:
;  208          0635  1  !
;  209          0636  1
;  210          0637  1  REQUIRE 'RTLML:OTSISB';                          ! get length of ISB
;  211          0805  1
;  212          0806  1  REQUIRE 'RTLML:OTSLUB';                          ! get length of LUB
;  213          0946  1
;  214          0947  1  REQUIRE 'RTLIN:RTLPSECT';                        ! Define DECLARE_PSECTs macro
;  215          1042  1
;  216          1043  1  REQUIRE 'RTLIN:OTSCCBREQ';                       ! Define interface to  OTS$PUSH_CCB
;  217          1141  1
;  218          1142  1  LIBRARY 'RTLSTARLE';                             ! STARLET library for macros and symbols
;  219          1143  1
;  220          1144  1  !
;  221          1145  1  ! MACROS:
;  222          1146  1  !
;  223          1147  1
;  224          1148  1  MACRO
;  225        M 1149  1      TEST_LUB_ISB =
;  226        M 1150  1  !+
;  227        M 1151  1  ! Give an error message if the ISB and the LUB overlap.  Try to make the
;  228        M 1152  1  ! message explicit enough to tell the maintainer exactly what to do, since
;  229        M 1153  1  ! it will print only when the RTL is being modified by someone who does not
;  230        M 1154  1  ! know about the LUB-ISB dependency, and therefore may need a lot of hand-
;  231        M 1155  1  ! holding.
;  232        M 1156  1  !-
;  233        M 1157  1
;  234        M 1158  1  %IF (LUB$K_NEG_BLN NEQ ISB$K_NEG_LUB)
;  235        M 1159  1  %THEN
;  236        M 1160  1
```

```
M  1 COMPILETIME
M  1     VAL1 = -ISB$K_NEG_LUB,
M  1     VAL2 = -LUB$K_NEG_BLN;
M  1
M  1 %ERROR (' LUB$K_NEG_BLN is not equal to ISB$K_NEG_LUB.',
M  1     '  This probably means that the LUB has been extended',
M  1     ' without editing the ISB to allow for it.  Please edit file OTSISB.MDL, making the -F,B,',
M  1     %NUMBER (VAL1), ' be -F,B,', %NUMBER (VAL2))
M  1 %FI
M  1
   1 %;
   1
   1 !
   1 ! EQUATED SYMBOLS:
   1 !
   1
   1 LITERAL
   1     K_TOTAL_CCB_LEN = LUB$K_LUB_LEN + ISB$K_ISB_LEN + RAB$C_BLN;        ! length of LUB+ISB+RAB
   1
   1 !
   1 ! PSECT DECLARATIONS:
   1 !
   1 DECLARE_PSECTS (OTS);                              ! declare PSECTs for OTS$ facility
   1 !
   1 ! OWN STORAGE:
   1 !
   1 !+
   1 ! The following quadword is the header of the I/O active queue.  Items
   1 ! are manipulated on this queue using the INSQUE and REMQUE instructions.
   1 !-
   1
   1 OWN
   1     OTS$Q_IO_ACTIVE : VECTOR [2];
   1
   1 !
   1 ! EXTERNAL REFERENCES:
   1 !
   1
   1 EXTERNAL ROUTINE
   1     LIB$GET_VM,                                    ! Allocate virtual memory
   1     LIB$FREE_VM,                                   ! Deallocate virtual memory
   1     LIB$STOP : NOVALUE,                            ! Signal a fatal error
   1     OTS$$FREE_LUN;                                 ! Deallocate a LUN
   1
   1 EXTERNAL LITERAL
   1     OTS$_FATINTERR : UNSIGNED (%BPVAL);            ! condition value for FATAL INTERNAL ERROR
   1
   1                                                    ! IN RUN-TIME LIBRARY error.
   1 !+
   1 ! The following externals represent the global part of the CCB
   1 ! data base.
   1 !-
   1
   1 EXTERNAL
   1     OTS$$V_CCB_INIT : VOLATILE,                    ! True if INIT done
   1     OTS$$AA_LUB_TAB : VOLATILE OTS$$LUB_TAB_ST     !
   1         [-LUB$K_ILUN_MIN + LUB$K_LUN_MAX + 1, LUB$K_ILUN_MIN],  ! Pointers to CCBs
```

```
;  294        1218  1       OTS$$V_IOINPROG : VOLATILE BITVECTOR,        ! True if LUN has I/O active
;  295        1219  1       OTS$$A_CUR_LUB,                              ! The current LUB
;  296        1220  1       OTS$$L_CUR_LUN,                              ! The current logical unit
;  297        1221  1       OTS$$L_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         PUSH$A_NEXT = [0, 0, %BPVAL, 0],         ! Next item
;  318          1241  1         PUSH$A_PREV = [4, 0, %BPVAL, 0],         ! Previous item
;  319          1242  1         PUSH$L_STS = [8, 0, %BPVAL, 0],          ! RMS status
;  320          1243  1         PUSH$L_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_IO_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
```

```
1 ROUTINE INITIALIZE : NOVALUE =                        ! Set up OWN storage
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Set up the LUB table, I/O Active queue and OTS$$L_CUR_LUN.
1 !       The LUB table and I/O active queue must be set up at run time
1 !       because they must be initialized with addresses, and this
1 !       cannot be done at link time or they will cease to be position
1 !       independent.  They must be initialized with addresses because
1 !       they are used by INSQUE and REMQUE to avoid disabling ASTs.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       IF (NOT .OTS$$V_CCB_INIT) THEN INITIALIZE ();
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       OTS$$AA_LUB_TAB
1 !       OTS$$Q_IO_ACTIVE
1 !       OTS$$V_IOINPROG
1 !       OTS$$L_CUR_LUN
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       OTS$$AA_LUB_TAB
1 !       OTS$$Q_IO_ACTIVE
1 !       OTS$$V_IOINPROG
1 !       OTS$$L_CUR_LUN
1 !       OTS$$V_CCB_INIT
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
1
2    BEGIN
2
2    LOCAL
2 !+
2 ! The following cell keeps track of whether or not ASTs were disabled
2 ! when we were called.
2 !-
2       AST_STATUS;
2
2 !+
2 ! First disable ASTs.  Then, if the initialization has not yet been
2 ! done, do it.  The initialization will have been done if an AST went
2 ! off between the test of OTS$$V_CCB_INIT and this point.
2 !-
2    AST_STATUS = $SETAST (ENBFLG = 0);
2
3    IF ( NOT .OTS$$V_CCB_INIT)
```

```
;  392          1314  2      THEN
;  393          1315  3          BEGIN
;  394          1316  3  !+
;  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 LUB$K_ILUN_MIN TO LUB$K_LUN_MAX DO
;  401          1323  3              IF (.OTS$$AA_LUB_TAB [.LUN, 0] EQL 0) THEN
;  402          1324  3              OTS$$AA_LUB_TAB [.LUN, 0] = OTS$$AA_LUB_TAB [.LUN, 1] = OTS$$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          OTS$Q_IO_ACTIVE [0] = OTS$Q_IO_ACTIVE [1] = OTS$Q_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          OTS$$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 EQL SS$_WASSET) THEN $SETAST (ENBFLG = 1);
;  419          1341  2
;  420          1342  2      RETURN;
;  421          1343  1      END;                                        ! of routine INITIALIZE


                                                                .TITLE  OTS$$CCB
                                                                .IDENT  \1-057\

                                                                .PSECT  _OTS$DATA,NOEXE,  PIC,2

                                                  00000 OTS$Q_IO_ACTIVE:
                                                                .BLKB   8

                                                                .EXTRN  LIB$GET_VM, LIB$FREE_VM
                                                                .EXTRN  LIB$STOP, OTS$$FREE_LUN
                                                                .EXTRN  OTS$_FATINTERR, OTS$$V_CCB_INIT
                                                                .EXTRN  OTS$$AA_LUB_TAB
                                                                .EXTRN  OTS$$V_IOINPROG
                                                                .EXTRN  OTS$$A_CUR_LUB, OTS$$L_CUR_LUN
                                                                .EXTRN  OTS$$L_LVL_CTR, SYS$SETAST

                                                                .PSECT  _OTS$CODE,NOWRT,  SHR,  PIC,2

                                             003C 00000 INITIALIZE:
                                                                .WORD   Save R2,R3,R4,R5                        ; 1257
                          55 00000000G    00   9E 00002         MOVAB   SYS$SETAST, R5                          ;
                          54 00000000G    00   9E 00009         MOVAB   OTS$$V_CCB_INIT, R4                     ;
                          53 00000000'    EF   9E 00010         MOVAB   OTS$Q_IO_ACTIVE, R3                     ;
                                          7E   D4 00017         CLRL    -(SP)                                   ; 1311
                          65              01   FB 00019         CALLS   #1, SYS$SETAST                          ;
                          31               64  E8 0001C            BLBS    OTS$$V_CCB_INIT, 3$                          ; 1313
                          51               08  CE 0001F            MNEGL   #8, LUN                                      ; 1322
                          52 00000000G0041  7E 00022 1$:           MOVAQ   OTS$$AA_LUB_TAB+64[LUN], R2                  ; 1323
                                           62  D5 0002A            TSTL    (R2)                                         ;
                                           0D  12 0002C            BNEQ    2$                                           ;
                             00000000G0041  7F 0002E            PUSHAQ  OTS$$AA_LUB_TAB+68[LUN]                      ; 1324
                          9E               52  D0 00035            MOVL    R2, @(SP)+                                   ;
                          62               52  D0 00038            MOVL    R2, (R2)                                     ;
          DF              51 00000077      8F  F3 0003B 2$:        AOBLEQ  #119, LUN, 1$                                ; 1323
                          51               63  9E 00043            MOVAB   OTS$Q_IO_ACTIVE, R1                          ; 1329
                     04   A3               51  D0 00046            MOVL    R1, OTS$Q_IO_ACTIVE+4                        ;
                          63               51  D0 0004A            MOVL    R1, OTS$Q_IO_ACTIVE                          ;
                          64               01  D0 0004D            MOVL    #1, OTS$$V_CCB_INIT                          ; 1333
                          09               50  D1 00050 3$:        CMPL    AST_STATUS, #9                               ; 1340
                                           05  12 00053            BNEQ    4$                                           ;
                                           01  DD 00055            PUSHL   #1                                           ;
                          65               01  FB 00057            CALLS   #1, SYS$SETAST                               ;
                                               04 0005A 4$:        RET                                                  ; 1343

; Routine Size:  91 bytes,    Routine Base:  _OTS$CODE + 0000
```

```
1 ROUTINE PUSH_FAKE : CALL_CCB =                  ! Push a "fake" active record
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Push onto the I/O Active queue a place holder.  This is to
1 !       satisfy POP_ACTIVE when we can't actually push the CCB.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       CALL PUSH_FAKE ();
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       OTS$$Q_IO_ACTIVE
1 !       OTS$$L_CUR_LUN
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       OTS$$Q_IO_ACTIVE        Holds previous I/O on this LUN
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Calls LIB$GET_VM to get virtual memory.
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         CCB : REF BLOCK [, BYTE];
2
2     LOCAL
2 !+
2 ! Declare the pointer to the block to push.
2 !-
2         PUSH : REF BLOCK [PUSH$K_LENGTH, BYTE] FIELD (PUSH_ITEM),
2         LUN;
2
2     LUN = .OTS$$L_CUR_LUN;
2 !+
2 ! Get virtual memory to hold the fake activation record.
2 !-
2     BEGIN
3
3     LOCAL
3         GET_VM_RESULT;
3
3     GET_VM_RESULT = LIB$GET_VM (%REF (PUSH$K_LENGTH), PUSH);
3
3     IF ( NOT .GET_VM_RESULT) THEN RETURN (OTS$K_PUSH_FAIL);
3
2     END;
2 !+
;   480          1401  2 ! Copy the old LUN into the fake record, and mark it as fake.
;   481          1402  2 !-
;   482          1403  2      PUSH [PUSH$W_LUN] = .LUN;
;   483          1404  2      PUSH [PUSH$V_FAKE] = 1;
;   484          1405  2 !+
;   485          1406  2 ! Put this item on the I/O Active list.
;   486          1407  2 !-
;   487          1408  2      INSQUE (.PUSH, OTS$Q_IO_ACTIVE);
;   488          1409  2 !+
;   489          1410  2 ! We also set OTS$$L_CUR_LUN to LUB$K_LUN_MAX+1 to prevent an
;   490          1411  2 ! AST from pushing that LUB again.  An extra push before this point
;   491          1412  2 ! does not cause any harm (only wastes a little time).
;   492          1413  2 !-
;   493          1414  2      OTS$$L_CUR_LUN = LUB$K_LUN_MAX + 1;
;   494          1415  2      RETURN (OTS$K_PUSH_OK);
;   495          1416  1      END;                                    ! of routine PUSH_FAKE



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

; Routine Size:  68 bytes,    Routine Base:  _OTS$CODE + 005B
```

```
1 ROUTINE PUSH_ACTIVE (LOGICAL_UNIT,                    ! The new LUN
1         RECURSIVE_IO                                  ! True if really recursive I/O
1     ) : CALL_CCB =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Place the ISB, etc. of the currently active logical unit on the
1 !       I/O Active queue so that another I/O statement may be started.
1 !       The I/O statement to be started may be on the same or another
1 !       logical unit as the one being interrupted.  When the new I/O
1 !       statement is complete the old one will be continued, so the I/O
1 !       active queue has a first-in-first-out discipline.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       RESULT = CALL PUSH_ACTIVE (LUN, RECURSIVE_IO);
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       LOGICAL_UNIT.rl.v       The new LUN
1 !       RECURSIVE_IO.rl.v       True if this LUN was already active
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       OTS$$AA_LUB_TAB
1 !       OTS$$Q_IO_ACTIVE
1 !       OTS$$L_CUR_LUN
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       OTS$$Q_IO_ACTIVE        Holds previous I/O on this LUN
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Calls LIB$GET_VM to get virtual memory.
1 !--
1
2   BEGIN
2
2   EXTERNAL REGISTER
2       CCB : REF BLOCK [, BYTE];
2
2   LOCAL
2 !+
2 ! Declare the pointer to the block to push.
2 !-
2       PUSH : REF BLOCK [PUSH$K_LENGTH, BYTE] FIELD (PUSH_ITEM);
2
2 !+
2 ! If there is no need to push anything, push a fake activation
2 ! record to satisfy POP_ACTIVE.
2 !-
2
2   IF (.OTS$$L_CUR_LUN GTR LUB$K_LUN_MAX) THEN RETURN (PUSH_FAKE ());
2
2 !+
! Check for this being an AST between the clearing of OTS$$V_IOINPROG
! and the setting of OTS$$L_CUR_LUN to LUB$K_LUN_MAX + 1.  If it
! is we cannot push the CCB, since, with RECURSIVE_IO clear,
! OTS$$V_IOINPROG will be cleared before the call to POP_ACTIVE,
! and we might try to pop into a deallocated CCB.
!-

    IF ((.OTS$$L_CUR_LUN EQL .LOGICAL_UNIT) AND ( NOT .RECURSIVE_IO)) THEN RETURN (PUSH_FAKE ());

    CCB = .OTS$$AA_LUB_TAB [.OTS$$L_CUR_LUN, 0];
!+
! If the queue is empty then the deallocation code has removed the LUB
! from the LUB table but has not yet popped OTS$$L_CUR_LUN.  Since
! the deallocation code will finish its deallocation no matter what
! we do here we need not push anything.  If any I/O is tried to this
! LUN it will create a new LUB.  The recursive flag may be set
! needlessly,  but that will only cause a problem in languages which
! do not support recursive I/O, and, actually, the higher I/O has not
! quite finished yet, so that is OK.
!-

    IF (.CCB EQLA OTS$$AA_LUB_TAB [.OTS$$L_CUR_LUN, 0]) THEN RETURN (PUSH_FAKE ());

!+
! The LUB is still allocated, do some consistency checks.
! We cannot check OTS$$AA_CUR_LUB since we may be in an AST that
! occurred after the update of OTS$$AA_CUR_LUB but before OTS$$L_CUR_LUN.
!-
    CCB = .CCB + (.CCB - CCB [LUB$Q_QUEUE]);

    IF (.CCB [LUB$W_LUN] NEQ .OTS$$L_CUR_LUN) THEN LIB$STOP (OTS$_FATINTERR);

!+
! Get virtual memory to hold the old ISB, etc.
!-
    BEGIN

    LOCAL
        GET_VM_RESULT;

    GET_VM_RESULT = LIB$GET_VM (%REF (PUSH$K_LENGTH), PUSH);

    IF ( NOT .GET_VM_RESULT) THEN RETURN (OTS$K_PUSH_FAIL);

    END;
!+
! Make sure there is no RMS I/O active on the RAB.
!-

    IF (.RECURSIVE_IO) THEN $WAIT (RAB = .CCB);

!+
! Copy the ISB and a few other things that need to be preserved
! over recursive I/O into the block we just allocated.
!-
    CH$MOVE (ISB$K_ISB_LEN, .CCB - ISB$K_ISB_LEN - LUB$K_LUB_LEN, PUSH [PUSH$X_ISB]);
    PUSH [PUSH$V_PMT] = .CCB [RAB$V_PMT];
```

```
;  611        1531  2
;  612        1532  3      IF (.PUSH [PUSH$V_PMT])
;  613        1533  2      THEN
;  614        1534  3          BEGIN
;  615        1535  3          CH$MOVE (.CCB [RAB$B_PSZ], .CCB [RAB$L_PBF], PUSH [PUSH$T_PROMPT]);
;  616        1536  3          PUSH [PUSH$B_PSZ] = .CCB [RAB$B_PSZ];
;  617        1537  2          END;
;  618        1538  2
;  619        1539  2      PUSH [PUSH$B_TMO] = .CCB [RAB$B_TMO];
;  620        1540  2      PUSH [PUSH$L_STS] = .CCB [RAB$L_STS];
;  621        1541  2      PUSH [PUSH$L_STV] = .CCB [RAB$L_STV];
;  622        1542  2      PUSH [PUSH$V_IO_ACT] = .CCB [LUB$V_IO_ACTIVE];
;  623        1543  2      PUSH [PUSH$V_FAKE] = 0;
;  624        1544  2  !+
;  625        1545  2  ! Record the logical unit number so that POP_ACTIVE knows where to
;  626        1546  2  ! restore this item when it is popped.
;  627        1547  2  !-
;  628        1548  2      PUSH [PUSH$W_LUN] = .CCB [LUB$W_LUN];
;  629        1549  2  !+
;  630        1550  2  ! Put this item on the I/O Active list.
;  631        1551  2  !-
;  632        1552  2      INSQUE (.PUSH, OTS$Q_IO_ACTIVE);
;  633        1553  2  !+
;  634        1554  2  ! That LUB is no longer the active one, mark it so.
;  635        1555  2  !-
;  636        1556  2      CCB [LUB$V_IO_ACTIVE] = 0;
;  637        1557  2  !+
;  638        1558  2  ! We also set OTS$$L_CUR_LUN to LUB$K_LUN_MAX+1 to prevent an
;  639        1559  2  ! AST from pushing that LUB again.  An extra push before this point
;  640        1560  2  ! does not cause any harm (only wastes a little time).
;  641        1561  2  !-
;  642        1562  2      OTS$$L_CUR_LUN = LUB$K_LUN_MAX + 1;
;  643        1563  2      RETURN (OTS$K_PUSH_OK);
;  644        1564  1      END;                                        ! of routine PUSH_ACTIVE



                                                                  .EXTRN  SYS$WAIT

                                            00FC 00000 PUSH_ACTIVE:
                                                                  .WORD   Save R2,R3,R4,R5,R6,R7              ; 1417
                     57 0000000G    00   9E 00002                 MOVAB   OTS$$L_CUR_LUN, R7                  ;
                     5E             08   C2 00009                 SUBL2   #8, SP                              ;
           00000077  8F             67   D1 0000C                 CMPL    OTS$$L_CUR_LUN, #119                ; 1471
                                    1D   14 00013                 BGTR    2$                                  ;
                 04  AC             67   D1 00015                 CMPL    OTS$$L_CUR_LUN, LOGICAL_UNIT        ; 1481
                                    04   12 00019                 BNEQ    1$                                  ;
                     13       08    AC   E9 0001B                 BLBC    RECURSIVE_IO, 2$                    ;
                     50             67   D0 0001F 1$:             MOVL    OTS$$L_CUR_LUN, R0                  ; 1483
                     50 0000000G0040     7E 00022                 MOVAQ   OTS$$AA_LUB_TAB+64[R0], R0          ;
                     5B             60   D0 0002A                 MOVL    (R0), CCB                           ;
                     50             5B   D1 0002D                 CMPL    CCB, R0                             ; 1495
                                    05   12 00030                 BNEQ    3$                                  ;
                 86  AF             00   FB 00032 2$:             CALLS   #0, PUSH_FAKE                       ;
                                         04 00036                 RET                                         ;
       50            5B             5B   C3 00037 3$:             SUBL3   CCB, CCB, R0                        ; 1502
                     5B       58 A04B    9E 0003B                 MOVAB   88(R0)[CCB], CCB                    ;
```

```
        67      C6  AB              10            00 EC 00040       CMPV     #0, #16, -58(CCB), OTS$$L_CUR_LUN        ; 1504
                                    0D            13 00046          BEQL     4$                                       ;
                                       00000000G  8F DD 00048       PUSHL    #OTS$_FATINTERR                          ;
                        00000000G   00            01 FB 0004E       CALLS    #1, LIB$STOP                             ;
                                           04     AE 9F 00055 4$:   PUSHAB   PUSH                                     ; 1514
                      04    AE           0121     8F 3C 00058       MOVZWL   #289, 4(SP)                              ;
                                           04     AE 9F 0005E       PUSHAB   4(SP)                                    ;
                        00000000G   00            02 FB 00061       CALLS    #2, LIB$GET_VM                           ;
                                    04            50 E8 00068       BLBS     GET_VM_RESULT, 5$                        ; 1516
                                    50            03 D0 0006B       MOVL     #3, R0                                   ;
                                                     04 0006E       RET                                               ;
                                    09     08     AC E9 0006F 5$:   BLBC     RECURSIVE_IO, 6$                         ; 1523
                                                  5B DD 00073       PUSHL    CCB                                      ;
                        00000000G   00            01 FB 00075       CALLS    #1, SYS$WAIT                             ;
                                    56     04     AE D0 0007C 6$:   MOVL     PUSH, R6                                 ; 1529
                65  A6       FEE0   CB   00BC     8F 28 00080       MOVC3    #188, -288(CCB), 101(R6)                 ;
        50      07  AB              01            06 EF 00089       EXTZV    #6, #1, 7(CCB), R0                       ; 1530
   14   A6          01              02            50 F0 0008F       INSV     R0, #2, #1, 20(R6)                       ;
                    0F         14   A6            02 E1 00095       BBC      #2, 20(R6), 7$                           ; 1532
                                    50     34     AB 9A 0009A       MOVZBL   52(CCB), R0                              ; 1535
                15  A6         30   BB            50 28 0009E       MOVC3    R0, @48(CCB), 21(R6)                     ;
                               12   A6     34     AB 90 000A4       MOVB     52(CCB), 18(R6)                          ; 1536
                               13   A6     1F     AB 90 000A9 7$:   MOVB     31(CCB), 19(R6)                          ; 1539
                               08   A6     08     AB 7D 000AE       MOVQ     8(CCB), 8(R6)                            ; 1540
        50      FC  AB              01            01 EF 000B3       EXTZV    #1, #1, -4(CCB), R0                      ; 1542
   14   A6          01              00            50 F0 000B9       INSV     R0, #0, #1, 20(R6)                       ;
                               14   A6            02 8A 000BF       BICB2    #2, 20(R6)                               ; 1543
                               10   A6     C6     AB B0 000C3       MOVW     -58(CCB), 16(R6)                         ; 1548
                        00000000'   EF            66 0E 000C8       INSQUE   (R6), OTS$$Q_IO_ACTIVE                   ; 1552
                               FC   AB            02 8A 000CF       BICB2    #2, -4(CCB)                              ; 1556
                               67          78     8F 9A 000D3       MOVZBL   #120, OTS$$L_CUR_LUN                     ; 1562
                               50                 01 D0 000D7       MOVL     #1, R0                                   ; 1563
                                                     04 000DA       RET                                               ; 1564

; Routine Size:  219 bytes,    Routine Base:  _OTS$CODE + 009F
```

```
1 ROUTINE ALLOCATE (LOGICAL_UNIT) : CALL_CCB =     ! Allocate LUB/ISB/RAB
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Allocate the LUB/ISB/RAB for this logical unit, watching out for
1 !       ASTs which may do the allocation as we are running.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       CALL ALLOCATE (.LOGICAL_UNIT)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       LOGICAL_UNIT.rl.v        The logical unit number for this CCB
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       OTS$$AA_LUB_TAB
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       OTS$$AA_LUB_TAB
1 !       CCB
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Calls LIB$GET_VM to get virtual memory.
1 !       May call LIB$FREE_VM to free that same virtual memory.
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         CCB : REF BLOCK [, BYTE];
2
2     LOCAL
2         INSQUE_ADDR,                              ! Address for INSQUE instruction
2         REMQUE_ADDR,                              ! Address for REMQUE instruction
2         CCB_ADDR;                                 ! Address of the allocated CCB
2
2 !+
2 ! Test the definitions of the LUB and ISB for consistency.  This is
2 ! purely a compile-time test; it generates no code.
2 !-
2     TEST_LUB_ISB;
2 !+
2 ! We must allocate.  This case is a little complex since an AST may
2 ! allocate the LUB.  We handle this by preparing the LUB and then
2 ! checking to see if an AST allocated one.  If so, we deallocate ours.
2 !-
3     BEGIN
3
3     LOCAL
3         GET_VM_RESULT;
3
3     GET_VM_RESULT = LIB$GET_VM (%REF (K_TOTAL_CCB_LEN), CCB_ADDR);
```

```
3
3      IF ( NOT .GET_VM_RESULT) THEN RETURN (OTS$K_PUSH_FAIL);
3
2      END;
2  !+
2  ! Clear the newly allocated LUN and RAB (but not ISB).  Adjust the
2  ! contents of the control block pointer (CCB) so that it points to
2  ! the beginning of the RAB.  (The ISB and LUB preceed the RAB using
2  ! negative offsets with respect to register CCB.)
2  ! Set the unit number in the newly allocated LUB.
2  !-
2      CCB = .CCB_ADDR;
2      CH$FILL (0, LUB$K_LUB_LEN + RAB$C_BLN, .CCB + ISB$K_ISB_LEN);
2      CCB = .CCB + ISB$K_ISB_LEN + LUB$K_LUB_LEN;
2      CCB [LUB$W_LUN] = .LOGICAL_UNIT;
2  !+
2  ! Initialize RAB to constants which never change.
2  ! Block ID, block length, and bit to make $PUT do $UPDATE if
2  ! record exists.  Also truncate on sequential $PUT not at EOF.
2  ! Note: TPT bit depends on FOP TRN bit being set in order to take effect.
2  ! Set read-ahead, write-behind and locate mode for GETs.
2  !-
2      CCB [RAB$B_BID] = RAB$C_BID;
2      CCB [RAB$B_BLN] = RAB$C_BLN;
2      CCB [RAB$V_UIF] = 1;
2      CCB [RAB$V_TPT] = 1;
2      CCB [RAB$V_RAH] = 1;
2      CCB [RAB$V_WBH] = 1;
2      CCB [RAB$V_LOC] = 1;
2  !+
2  ! Set up LUB$A_BUDDY_PTR.   If this CCB is not its own buddy, this
2  ! field will be changed during open.
2  !-
2      CCB [LUB$A_BUDDY_PTR] = .CCB;
2  !+
2  ! See if an AST has allocated this LUB/RAB/ISB while we were preparing
2  ! ours above.  If so, we use the allocated one.  If the LUB was
2  ! allocated by an AST it cannot have I/O active, since the AST must
2  ! complete any I/O it starts.  In spite of this, it cannot be
2  ! deallocated because we have OTS$$V_IOINPROG set for the LUN.
2  !-
2      INSQUE_ADDR = OTS$$AA_LUB_TAB [.LOGICAL_UNIT, 1];
2
3      IF ( NOT INSQUE (CCB [LUB$Q_QUEUE], ..INSQUE_ADDR))
2      THEN
3          BEGIN
3  !+
3  ! This CCB is not the first in the queue, which means that an AST
3  ! has allocated one and put it in the queue before us.  Remove ours
3  ! and deallocate it.  We will use the LUB previously on the queue.
3  !-
3          REMQUE_ADDR = OTS$$AA_LUB_TAB [.LOGICAL_UNIT, 1];
3
3          CASE (REMQUE (..REMQUE_ADDR, CCB)) FROM 0 TO 3 OF
3              SET
3
3              [2] :
```

```
3  !+
3  ! Somebody removed the other entry.  This should never happen.
3  !-
3                 LIB$STOP (OTS$_FATINTERR);
3
3             [3] :
3  !+
3  ! The queue was empty.  This is unreasonable because OTS$$V_IOINPROG is set.
3  !-
3                 LIB$STOP (OTS$_FATINTERR);
3
3             [0] :
3  !+
3  ! All is well.  We can now free the CCB we just removed.
3  ! It had better be the one we allocated.
3  !-
3
4                 IF ((.CCB + (.CCB - CCB [LUB$Q_QUEUE])) NEQA (.CCB_ADDR + ISB$K_ISB_LEN + LUB$K_LUB_LEN))
3                 THEN
3                     LIB$STOP (OTS$_FATINTERR);
3
3             [INRANGE, OUTRANGE] :
3  !+
3  ! This should never happen;  the only possible values from the REMQUE
3  ! function are 0, 2 and 3.
3  !-
3                 LIB$STOP (OTS$_FATINTERR);
3             TES;
3
3  !+
3  ! Now free the LUB we allocated.
3  !-
4         BEGIN
4
4         LOCAL
4             FREE_VM_STATUS;
4
4         FREE_VM_STATUS = LIB$FREE_VM (%REF (K_TOTAL_CCB_LEN), CCB_ADDR);
4
4         IF ( NOT .FREE_VM_STATUS) THEN LIB$STOP (OTS$_FATINTERR);
4
3         END;
3  !+
3  ! Now fetch the CCB address.  It must still be there because of
3  ! OTS$$V_IOINPROG.
3  !-
3         CCB = .OTS$$AA_LUB_TAB [.LOGICAL_UNIT, 0];
3         CCB = .CCB + (.CCB - CCB [LUB$Q_QUEUE]);
2         END;
2
2     RETURN (OTS$K_PUSH_OK);
1     END;                                        ! of routine ALLOCATE
```

```
                                                                    01FC 00000 ALLOCATE:
                                                                                        .WORD    Save R2,R3,R4,R5,R6,R7,R8               ; 1565
                                                   58 00000000G   00   9E 00002         MOVAB    LIB$STOP, R8                            ;
                                                   57 00000000G   8F   D0 00009         MOVL     #OTS$_FATINTERR, R7                     ;
                                                   56 00000000G   00   9E 00010         MOVAB    OTS$$AA_LUB_TAB+68, R6                  ;
                                                   5E             08   C2 00017         SUBL2    #8, SP                                  ;
                                                             04   AE   9F 0001A         PUSHAB   CCB_ADDR                                ; 1621
                                              04   AE      0164   8F   3C 0001D         MOVZWL   #356, 4(SP)                             ;
                                                             04   AE   9F 00023         PUSHAB   4(SP)                                   ;
                                       00000000G   00             02   FB 00026         CALLS    #2, LIB$GET_VM                          ;
                                                   C4             50   E8 0002D         BLBS     GET_VM_RESULT, 1$                       ; 1623
                                                   50             03   D0 00030         MOVL     #3, R0                                  ;
                                                                       04 00033         RET                                              ;
                                                   5B        04   AE   D0 00034 1$:     MOVL     CCB_ADDR, CCB                           ; 1633
           00A8    8F              00              6E             00   2C 00038         MOVC5    #0, (SP), #0, #168, 188(CCB)            ; 1634
                                                           00BC   CB      0003F                                                          ;
                                                   5B      0120   CB   9E 00042         MOVAB    288(R11), CCB                           ; 1635
                                              C6   AB        04   AC   B0 00047         MOVW     LOGICAL_UNIT, -58(CCB)                  ; 1636
                                                   6B      4401   8F   B0 0004C         MOVW     #17409, (CCB)                           ; 1644
                                                   50        04   AB   9E 00051         MOVAB    4(CCB), R0                              ; 1646
                                                   60  00010612   8F   C8 00055         BISL2    #67090, (R0)                            ; 1650
                                              B8   AB             5B   D0 0005C         MOVL     CCB, -72(CCB)                           ; 1655
                                                   50        04   AC   D0 00060         MOVL     LOGICAL_UNIT, R0                        ; 1663
                                                   50           6640   7E 00064         MOVAQ    OTS$$AA_LUB_TAB+68[R0], INSQUE_ADDR     ;
                                              00   B0        A8   AB   0E 00068         INSQUE   -88(CCB), @0(INSQUE_ADDR)               ; 1665
                                                                  6C   13 0006D         BEQL     7$                                      ;
                                                   50        04   AC   D0 0006F         MOVL     LOGICAL_UNIT, R0                        ; 1673
                                                   51           6640   7E 00073         MOVAQ    OTS$$AA_LUB_TAB+68[R0], REMQUE_ADDR     ;
                                                   5B        00   B1   0F 00077         REMQUE   @0(REMQUE_ADDR), CCB                    ; 1675
                                                                  50   DC 0007B         MOVPSL   R0                                      ;
                   50              50              02             01   EF 0007D         EXTZV    #1, #2, R0, R0                          ;
                                   03              00             50   CF 00082         CASEL    R0, #0, #3                              ;
                 0021            0021            0021           000A      00086 2$:     .WORD    3$-2$,-                                 ;
                                                                                                 4$-2$,-                                 ;
                                                                                                 4$-2$,-                                 ;
                                                                                                 4$-2$                                   ;
                                                                  17   11 0008E         BRB      4$                                      ; 1688
                                   50              5B             5B   C3 00090 3$:     SUBL3    CCB, CCB, R0                            ; 1696
                                                   51        58 A04B   9E 00094         MOVAB    88(R0)[CCB], R1                         ;
                                   50         04   AE  00000120   8F   C1 00099         ADDL3    #288, CCB_ADDR, R0                      ;
                                                   50             51   D1 000A2         CMPL     R1, R0                                  ;
                                                                  05   13 000A5         BEQL     5$                                      ;
                                                                  57   DD 000A7 4$:     PUSHL    R7                                      ; 1698
                                                   68             01   FB 000A9         CALLS    #1, LIB$STOP                            ;
                                                             04   AE   9F 000AC 5$:     PUSHAB   CCB_ADDR                                ; 1716
                                              04   AE      0164   8F   3C 000AF         MOVZWL   #356, 4(SP)                             ;
                                                             04   AE   9F 000B5         PUSHAB   4(SP)                                   ;
                                       00000000G   00             02   FB 000B8         CALLS    #2, LIB$FREE_VM                         ;
                                                   05             50   E8 000BF         BLBS     FREE_VM_STATUS, 6$                      ; 1718
                                                                  57   DD 000C2         PUSHL    R7                                      ;
                                                   68             01   FB 000C4         CALLS    #1, LIB$STOP                            ;
                                                   50        04   AC   D0 000C7 6$:     MOVL     LOGICAL_UNIT, R0                        ; 1725
                                                             FC A640   7F 000CB         PUSHAQ   OTS$$AA_LUB_TAB+64[R0]                  ;
                                                   5B             9E   D0 000CF         MOVL     @(SP)+, CCB                             ;
                                   50              5B             5B   C3 000D2         SUBL3    CCB, CCB, R0                            ; 1726
                                                   5B        58 A04B   9E 000D6         MOVAB    88(R0)[CCB], CCB                        ;
                                                   50             01   D0 000DB 7$:     MOVL     #1, R0                                  ; 1729
                                       04 000DE          RET                                    ; 1730

; Routine Size:  223 bytes,    Routine Base:  _OTS$CODE + 017A

;  812          1731  1
```

```
1 GLOBAL ROUTINE OTS$$PUSH_CCB (                          ! Get a CCB, pushing old
1         LOGICAL_UNIT                                    ! Logical unit for this CCB
1 )     : JSB_CB_PUSH =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Load register CCB with a pointer to the LUB/ISB/RAB for this
1 !       logical unit.  If no LUB has been allocated, allocate one.
1 !       If there is already I/O active push down the old ISB, etc.
1 !       POP_ACTIVE will restore it.  We already know that this LUN is
1 !       not in use by FORTRAN.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       CALL OTS$$PUSH_CCB (logical_unit.rl.v)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       logical_unit.rl.v       Logical unit  - identifies CCB
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       OTS$$V_CCB_INIT
1 !       OTS$$AA_LUB_TAB
1 !       OTS$$Q_IO_ACTIVE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       CCB                     Set to adr. of allocated LUB/ISB/RAB
1 !       OTS$$Q_IO_ACTIVE        Holds previous I/O on this LUN
1 !       OTS$AA_LUB_TAB          Set of adr. of allocated LUB/ISB/RAB
1 !                               for logical_unit
1 !       LUB$W_LUN               Set to logical_unit
1 !       LUB$V_IO_ACTIVE         Set to indicate active I/O
1 !       OTS$$V_CCB_INIT         Always set to 1.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       May call LIB$GET_VM to get virtual memory.
1 !       In unusual cases, may call LIB$FREE_VM to free virtual memory.
1 !       The first time entered, calls INITIALIZE, which disables ASTs.
1 !--
1
2    BEGIN
2
2    EXTERNAL REGISTER
2       CCB : REF BLOCK [, BYTE];
2
2    LOCAL
2       RECURSIVE_IO;                                     ! =1 if we are doing recursive I/O
2
2 !+
2 ! If this is the first entry, call INITIALIZE to set up OWN storage.
2 ! Note that PUSH_CCB must be entered before POP_CCB, so this is the
2 ! first reference to this data base, except for FORTRAN, which is checked
2 ! for in INITIALIZE.
```

```
2 !-
2
2      IF ( NOT .OTS$$V_CCB_INIT) THEN INITIALIZE ();
2
2 !+
2 ! Count the level counter.  This must be done before the OTS$$V_IOINPROG
2 ! bit is set, otherwise an AST could find the OTS$$V_IOINPROG bit set but
2 ! level counter -1, which would mean that the PUSH and POP routines
2 ! would not be called and OTS$$V_IOINPROG would get cleared by the AST.
2 !-
2      OTS$$L_LVL_CTR = .OTS$$L_LVL_CTR + 1;
2 !+
2 ! Mark that this LUN has I/O active so that its LUB (if it has one yet)
2 ! will not be deallocated.  If it was already active, remember that.
2 !-
2      RECURSIVE_IO = (TESTBITSS (OTS$$V_IOINPROG [.LOGICAL_UNIT - LUB$K_ILUN_MIN]));
2 !+
2 ! If I/O is currently active, push the presently active unit.
2 !-
2
3      IF (.OTS$$L_LVL_CTR NEQ 0)
2      THEN
3          BEGIN
3
3          LOCAL
3              PUSH_RESULT;
3
3          PUSH_RESULT = PUSH_ACTIVE (.LOGICAL_UNIT, .RECURSIVE_IO);
3
3          IF (.PUSH_RESULT NEQ OTS$K_PUSH_OK) THEN RETURN (.PUSH_RESULT);
3
2          END;
2
2 !+
2 ! Allocate the LUB/ISB/RAB if necessary.  If an AST allocates it we
2 ! must release ours.  Note that, because OTS$$V_IOINPROG is set, if an
2 ! AST allocates the LUB it will not be deallocated.
2 !-
2      CCB = .OTS$$AA_LUB_TAB [.LOGICAL_UNIT, 0];
2
3      IF (.CCB NEQA OTS$$AA_LUB_TAB [.LOGICAL_UNIT, 0])
2      THEN
3          BEGIN
3 !+
3 ! The CCB is already allocated.  Adjust register CCB to point to it.
3 !-
3          CCB = .CCB + (.CCB - CCB [LUB$Q_QUEUE]);
3          END
2      ELSE
3          BEGIN
3
3          LOCAL
3              ALLOCATE_RESULT;
3
3          ALLOCATE_RESULT = ALLOCATE (.LOGICAL_UNIT);
3
3          IF (.ALLOCATE_RESULT NEQ OTS$K_PUSH_OK) THEN RETURN (.ALLOCATE_RESULT);
```

```
;  928          1846  3                 END;
;  929          1847  2         END;
;  930          1848  2
;  931          1849  2 !+
;  932          1850  2 ! Set OTS$$L_CUR_LUN to be the current logical unit number.  This is
;  933          1851  2 ! the cell that controls pushing.
;  934          1852  2 !-
;  935          1853  2     OTS$$L_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$V_IO_ACTIVE] = 1;
;  941          1859  2     CCB [ISB$V_RECURSIVE] = .RECURSIVE_IO;
;  942          1860  2 !+
;  943          1861  2 ! Set OTS$$A_CUR_LUB to point to the new current LUB.
;  944          1862  2 !-
;  945          1863  2     OTS$$A_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 [ISB$V_P_FORM_CH] = 0;
;  952          1870  2     CCB [ISB$V_DOLLAR] = 0;
;  953          1871  2     CCB [ISB$V_USER_ELEM] = 0;
;  954          1872  2     CCB [ISB$V_SLASH] = 0;
;  955          1873  2     CCB [ISB$V_LAST_REC] = 0;
;  956          1874  2     CCB [ISB$V_DE_ENCODE] = 0;
;  957          1875  2     CCB [ISB$V_LIS_HEAP] = 0;
;  958          1876  2 !+
;  959          1877  2 ! When we set OTS$$V_IOINPROG we tested it to see if I/O was already active
;  960          1878  2 ! on this LUN.  If it was we must return this information to our
;  961          1879  2 ! caller because some languages do not permit recursive I/O.
;  962          1880  2 !-
;  963          1881  2
;  964          1882  2     IF (.RECURSIVE_IO) THEN RETURN (OTS$K_PUSH_ACT);
;  965          1883  2
;  966          1884  2     RETURN (OTS$K_PUSH_OK);
;  967          1885  1     END;                                ! End of routine OTS$$PUSH_CCB



                                    52  DD 00000 OTS$$PUSH_CCB::
                                                         PUSHL   R2                                          ; 1732
                     5E             04  C2 00002         SUBL2   #4, SP                                      ;
                                    52  DD 00005         PUSHL   R2                                          ;
                     05 00000000G   00  E8 00007         BLBS    OTS$$V_CCB_INIT, 1$                         ; 1791
                FD94 CF             00  FB 0000E         CALLS   #0, INITIALIZE                              ;
                        00000000G   00  D6 00013 1$:     INCL    OTS$$L_LVL_CTR                              ; 1799
         52          6E             08  C1 00019         ADDL3   #8, LOGICAL_UNIT, R2                        ; 1804
                                    50  D4 0001D         CLRL    R0                                          ;
         02 00000000G   00          52  E3 0001F         BBCS    R2, OTS$$V_IOINPROG, 2$                     ;
                                    50  D6 00027         INCL    R0                                          ;
                04   AE             50  D0 00029 2$:     MOVL    R0, RECURSIVE_IO                            ;
```

```
                              00000000G  00  D5 0002D          TSTL    OTS$$L_LVL_CTR                          ; 1809
                                         10  13 00033          BEQL    3$                                      ;
                                  04     AE  DD 00035          PUSHL   RECURSIVE_IO                            ; 1816
                                  04     AE  DD 00038          PUSHL   LOGICAL_UNIT                            ;
                     FE06   CF           02  FB 0003B          CALLS   #2, PUSH_ACTIVE                         ;
                            01           50  D1 00040          CMPL    PUSH_RESULT, #1                         ; 1818
                                         52  12 00043          BNEQ    7$                                      ;
                            50 00000000G0042  7E 00045 3$:     MOVAQ   OTS$$AA_LUB_TAB[R2], R0                 ; 1827
                            5B           60  D0 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                        ;
                                         0C  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  D0 0006C 5$:      MOVL    LOGICAL_UNIT, OTS$$L_CUR_LUN            ; 1853
                       FC   AB           02  88 00073          BISB2   #2, -4(CCB)                             ; 1858
                            50      96   AB  9E 00077          MOVAB   -106(CCB), R0                           ; 1859
   01    A0           01    00      04   AE  F0 0007B          INSV    RECURSIVE_IO, #0, #1, 1(R0)             ;
                00000000G   00           5B  D0 00082          MOVL    CCB, OTS$$A_CUR_LUB                     ; 1863
                                         60  94 00089          CLRB    (R0)                                    ; 1875
                            05      04   AE  E9 0008B          BLBC    RECURSIVE_IO, 6$                        ; 1882
                            50           02  D0 0008F          MOVL    #2, R0                                  ;
                                         03  11 00092          BRB     7$                                      ;
                            50           01  D0 00094 6$:      MOVL    #1, R0                                  ; 1884
                            5E           08  C0 00097 7$:      ADDL2   #8, SP                                  ; 1885
                                         04  BA 0009A          POPR    #^M<R2>                                 ;
                                         05 0009C              RSB                                             ;

; Routine Size:  157 bytes,    Routine Base:  _OTS$CODE + 0259


;  968          1886  1
```

```
1 ROUTINE DEALLOCATE : CALL_CCB NOVALUE =          ! Deallocate LUB/ISB/RAB
  ;
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Deallocate the LUB/ISB/RAB for this logical unit, including
1 !       the allocated structures attached to it.  Also, deallocate the
1 !       LUN.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       CALL DEALLOCATE ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       OTS$$AA_LUB_TAB
1 !       CCB
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       OTS$$AA_LUB_TAB
1 !       CCB
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Calls LIB$FREE_VM to free virtual memory.
1 !--
1
2    BEGIN
2
2    EXTERNAL REGISTER
2        CCB : REF BLOCK [, BYTE];
2
2    LOCAL
2        REMQUE_ADDR,                             ! Address for REMQUE instruction
2        CCB_ADDR : REF BLOCK [0, BYTE],
2        BUDDY_CCB : REF BLOCK [0, BYTE],
2        LUN;
2
2 !+
2 ! We now deallocate the LUB/ISB/RAB.  An AST will not deallocate under
2 ! us because it will find OTS$$V_IOINPROG set at PUSH time, and will
2 ! therefore set ISB$V_RECURSIVE so as not to clear OTS$$V_IOINPROG at POP
2 ! time or deallocate the LUB.
2 !-
2    REMQUE_ADDR = OTS$$AA_LUB_TAB [.CCB [LUB$W_LUN], 0];
2
2    CASE (REMQUE (..REMQUE_ADDR, CCB_ADDR)) FROM 0 TO 3 OF
2        SET
2
2        [0, 3] :
2 !+
2 ! Zero means that there was more than one entry in the queue.
```

```
2 ! This implies that we have done a CLOSE in an AST which went off after
2 ! the INSQUE but before the compensating REMQUE in PUSH_CCB.
2 ! This should never happen because ISB$V_RECURSIVE
2 ! will be set in this case.
2 !
2 ! Three implies that there is nothing in the queue.
2 ! This means that an AST deallocated the LUB, which should not happen
2 ! because of the ISB$V_RECURSIVE test.
2 !-
2               LIB$STOP (OTS$_FATINTERR);
2
2           [2] :
2 !+
2 ! The queue is now empty.  This is correct.  We can now free the LUB.
2 ! Note that PUSH_CCB will allocate a new LUB if an AST goes off to it
2 ! here, and will carefully not push the LUB we are deallocating.
2 ! First perform a consistency check.
2 !-
2
2               IF (.CCB_ADDR + (.CCB_ADDR - CCB_ADDR [LUB$Q_QUEUE]) NEQA .CCB) THEN LIB$STOP (OTS$_FATINTERR);
2
2           [INRANGE, OUTRANGE] :
2 !+
2 ! This should never happen. The only possible values of REMQUE are
2 ! 0, 2 and 3.
2 !-
2               LIB$STOP (OTS$_FATINTERR);
2           TES;
2
2 !+
2 ! Since the LUB/ISB/RAB can no longer be used, clear its OTS$$V_IOINPROG
2 ! bit.  An AST after this point will not indicate recursive I/O.
2 !-
2
2       IF (TESTBITCC (OTS$$V_IOINPROG [.CCB [LUB$W_LUN] - LUB$K_ILUN_MIN])) THEN LIB$STOP (OTS$_FATINTERR);
2
2 !+
2 ! Clear this LUN's buddy's buddy pointer, which points to us.
2 !-
2       BUDDY_CCB = .CCB [LUB$A_BUDDY_PTR];
2
2       IF (.BUDDY_CCB NEQA 0) THEN BUDDY_CCB [LUB$A_BUDDY_PTR] = 0;
2
2 !+
2 ! Free the record buffer if we allocated it.
2 !-
2
2       IF (( NOT .CCB [LUB$V_USER_RBUF]) AND (.CCB [LUB$A_UBF] NEQA 0))
2       THEN
3           BEGIN
3
3           LOCAL
3               FREE_VM_STATUS;
3
3           FREE_VM_STATUS = LIB$FREE_VM (%REF (.CCB [LUB$W_RBUF_SIZE]), CCB [LUB$A_UBF]);
3
3           IF ( NOT .FREE_VM_STATUS) THEN LIB$STOP (OTS$_FATINTERR);
```

```
3
2           END;
2
2    !+
2    ! Free the file name string, if it is allocated.
2    !-
2
3       IF (.CCB [LUB$V_VIRT_RSN])
2       THEN
3           BEGIN
3
3           LOCAL
3               FREE_VM_STATUS;
3
3           FREE_VM_STATUS = LIB$FREE_VM (%REF (.CCB [LUB$B_RSL]), CCB [LUB$A_RSN]);
3
3           IF ( NOT .FREE_VM_STATUS) THEN LIB$STOP (OTS$_FATINTERR);
3
3           CCB [LUB$V_VIRT_RSN] = 0;
2           END;
2
2    !+
2    ! Free the prompt buffer, if there is one.
2    !-
3       BEGIN
3
3       LOCAL
3           FREE_VM_STATUS;
3
4       IF ((.CCB [RAB$L_PBF] NEQA 0) AND (.CCB [RAB$V_PMT]))
3       THEN
4           BEGIN
4           FREE_VM_STATUS = LIB$FREE_VM (%REF (LUB$K_PBUF_SIZ), CCB [RAB$L_PBF]);
4
4           IF ( NOT .FREE_VM_STATUS) THEN LIB$STOP (OTS$_FATINTERR);
4
3           END;
3
2       END;
2    !+
2    ! Remember the logical unit number, since we will need it in a minute.
2    !-
2       LUN = .CCB [LUB$W_LUN];
2    !+
2    ! Now, at last, we can free the CCB itself.
2    !-
3       BEGIN
3
3       LOCAL
3           FREE_VM_STATUS;
3
3       FREE_VM_STATUS = LIB$FREE_VM (%REF (K_TOTAL_CCB_LEN), %REF (.CCB - ISB$K_ISB_LEN - LUB$K_LUB_LEN));
3
3       IF ( NOT .FREE_VM_STATUS) THEN LIB$STOP (OTS$_FATINTERR);
3
2       END;
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
: '148          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 OTS$$FREE_LUN (LUN)) THEN LIB$STOP (OTS$_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
                         55 00000000G   00   9E 00002             MOVAB   LIB$FREE_VM, R5                           ;
                         54 00000000G   00   9E 00009             MOVAB   LIB$STOP, R4                              ;
                         53 00000000G   8F   D0 00010             MOVL    #OTS$_FATINTERR, R3                       ;
                         5E             0C   C2 00017             SUBL2   #12, SP                                   ;
                         50        C6   AB   32 0001A             CVTWL   -58(CCB), R0                              ; 1936
                         51 00000000G0040    7E 0001E             MOVAQ   OTS$$AA_LUB_TAB+64[R0], REMQUE_ADDR       ;
                         52        00   B1   0F 00026             REMQUE  @0(REMQUE_ADDR), CCB_ADDR                 ; 1938
                                        50   DC 0002A             MOVPSL  R0                                        ;
   50          50        02             01   EF 0002C             EXTZV   #1, #2, R0, R0                            ;
               03        00             50   CF 00031             CASEL   R0, #0, #3                                ;
 0018        000A      0018           0018      00035 1$:         .WORD   3$-1$,-                                   ;
                                                                          3$-1$,-                                   ;
                                                                          2$-1$,-                                   ;
                                                                          3$-1$                                     ;
                                        0E   11 0003D             BRB     3$                                        ; 1953
               50        52             52   C3 0003F 2$:         SUBL3   CCB_ADDR, CCB_ADDR, R0                    ; 1963
                         50      58 A042     9E 00043             MOVAB   88(R0)[CCB_ADDR], R0                      ;
                         5B             50   D1 00048             CMPL    R0, CCB                                   ;
                                        05   13 0004B             BEQL    4$                                        ;
                                        53   DD 0004D 3$:         PUSHL   R3                                        ;
                         64             01   FB 0004F             CALLS   #1, LIB$STOP                              ;
                         50        C6   AB   32 00052 4$:         CVTWL   -58(CCB), R0                              ; 1978
                         50             08   C0 00056             ADDL2   #8, R0                                    ;
               05 00000000G  00         50   E4 00059             BBSC    R0, OTS$$V_IOINPROG, 5$                   ;
                                        53   DD 00061             PUSHL   R3                                        ;
                         64             01   FB 00063             CALLS   #1, LIB$STOP                              ;
                         50        B8   AB   D0 00066 5$:         MOVL    -72(CCB), BUDDY_CCB                       ; 1983
                                        03   13 0006A             BEQL    6$                                        ; 1985
                                   B8   A0   D4 0006C             CLRL    -72(BUDDY_CCB)                            ;
                                   FF   AB   95 0006F 6$:         TSTB    -1(CCB)                                   ; 1991
```

```
                              1B  19 00072          BLSS    7$                               ;
                        9C    AB  D5 00074          TSTL    -100(CCB)                        ;
                              16  13 00077          BEQL    7$                               ;
                        9C    AB  9F 00079          PUSHAB  -100(CCB)                        ; 1998
             08    AE   D2    AB  3C 0007C          MOVZWL  -46(CCB), 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$STOP                     ;
                   1A   FE    AB  E9 0008F 7$:      BLBC    -2(CCB), 9$                      ; 2008
                        F8    AB  9F 00093          PUSHAB  -8(CCB)                          ; 2015
             08    AE   F7    AB  9A 00096          MOVZBL  -9(CCB), 8(SP)                   ;
                        08    AE  9F 0009B          PUSHAB  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$STOP                     ;
             FE    AB         01  8A 000A9 8$:      BICB2   #1, -2(CCB)                      ; 2019
                        30    AB  D5 000AD 9$:      TSTL    48(CCB)                          ; 2030
                              1B  13 000B0          BEQL    10$                              ;
    16       07    AB         06  E1 000B2          BBC     #6, 7(CCB), 10$                  ;
                        30    AB  9F 000B7          PUSHAB  48(CCB)                          ; 2033
             08    AE   50    8F  9A 000BA          MOVZBL  #80, 8(SP)                       ;
                        08    AE  9F 000BF          PUSHAB  8(SP)                            ;
                   65         02  FB 000C2          CALLS   #2, LIB$FREE_VM                  ;
                   05         50  E8 000C5          BLBS    FREE_VM_STATUS, 10$              ; 2035
                              53  DD 000C8          PUSHL   R3                               ;
                   64         01  FB 000CA          CALLS   #1, LIB$STOP                     ;
             08    AE   C6    AB  32 000CD 10$:     CVTWL   -58(CCB), LUN                    ; 2043
             04    AE FEE0    CB  9E 000D2          MOVAB   -288(R11), 4(SP)                 ; 2052
                        04    AE  9F 000D8          PUSHAB  4(SP)                            ;
             04    AE 0164    8F  3C 000DB          MOVZWL  #356, 4(SP)                      ;
                        04    AE  9F 000E1          PUSHAB  4(SP)                            ;
                   65         02  FB 000E4          CALLS   #2, LIB$FREE_VM                  ;
                   05         50  E8 000E7          BLBS    FREE_VM_STATUS, 11$              ; 2054
                              53  DD 000EA          PUSHL   R3                               ;
                   64         01  FB 000EC          CALLS   #1, LIB$STOP                     ;
                              5B  D4 000EF 11$:     CLRL    CCB                              ; 2061
                        08    AE  D5 000F1          TSTL    LUN                              ; 2071
                              12  19 000F4          BLSS    12$                              ;
                        08    AE  9F 000F6          PUSHAB  LUN                              ; 2074
       00000000G   00         01  FB 000F9          CALLS   #1, OTS$$FREE_LUN                ;
                   05         50  E8 00100          BLBS    R0, 12$                          ;
                              53  DD 00103          PUSHL   R3                               ;
                   64         01  FB 00105          CALLS   #1, LIB$STOP                     ;
                                  04 00108 12$:     RET                                      ; 2077

; Routine Size:  265 bytes,    Routine Base:  _OTS$CODE + 02F6
```

```
1 ROUTINE POP_ACTIVE : CALL_CCB NOVALUE =         ! Pop old active unit
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Restore the status of an interrupted I/O statement using the
1 !       information saved when the statement was interrupted.  All of
1 !       the ISB is restored, and a few other things.  In some unusual
1 !       cases there is no CCB to restore to, so only OTS$$L_CUR_LUN is
1 !       restored.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       CALL POP_ACTIVE ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       The DEALLOC bit in the LUB
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       The ISB, and some other fields of the CCB
1 !       CCB             The restored CCB
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Calls LIB$FREE_VM to free virtual memory.
1 !--
1
2    BEGIN
2
2    EXTERNAL REGISTER
2        CCB : REF BLOCK [, BYTE];
2
2    LOCAL
2        PUSH : REF BLOCK [PUSH$K_LENGTH, BYTE] FIELD (PUSH_ITEM),
2        LUN;                                   ! Logical unit number being restored
2
2 !+
2 ! Get an activation record off the I/O Active queue.  It had better
2 ! be there.
2 !-
2
2    IF (REMQUE (.OTS$Q_IO_ACTIVE [0], PUSH)) THEN LIB$STOP (OTS$_FATINTERR);
2
2 !+
2 ! Fetch the logical unit number associated with this record.
2 !-
2    LUN = .PUSH [PUSH$W_LUN];
2 !+
2 ! If this is a fake activation record, just store the LUN.
2 !-
2
```

```
; 1219          2135  3       IF (.PUSH [PUSH$V_FAKE])
; 1220          2136  2       THEN
; 1221          2137  2           OTS$$L_CUR_LUN = .LUN
; 1222          2138  2       ELSE
; 1223          2139  3           BEGIN
; 1224          2140  3   !+
; 1225          2141  3   ! If this LUN does not have I/O in progress then something is very
; 1226          2142  3   ! wrong.
; 1227          2143  3   !-
; 1228          2144  3
; 1229          2145  3           IF ( NOT .OTS$$V_IOINPROG [.LUN - LUB$K_ILUN_MIN]) THEN LIB$STOP (OTS$_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 OTS$$L_CUR_LUN before copying
; 1234          2150  3   ! data from the I/O Active entry because only the LUN indicated by
; 1235          2151  3   ! OTS$$L_CUR_LUN will get pushed.
; 1236          2152  3   !-
; 1237          2153  3           OTS$$L_CUR_LUN = .LUN;
; 1238          2154  3           CCB = .OTS$$AA_LUB_TAB [.LUN, 0];
; 1239          2155  3           CCB = .CCB + (CCB [0, 0, 0, 0] - CCB [LUB$Q_QUEUE]);
; 1240          2156  3           OTS$$A_CUR_LUB = .CCB;
; 1241          2157  3           CCB [LUB$V_IO_ACTIVE] = .PUSH [PUSH$V_IO_ACT];
; 1242          2158  3           CCB [RAB$L_STS] = .PUSH [PUSH$L_STS];
; 1243          2159  3           CCB [RAB$L_STV] = .PUSH [PUSH$L_STV];
; 1244          2160  3           CCB [RAB$B_TMO] = .PUSH [PUSH$B_TMO];
; 1245          2161  3
; 1246          2162  4           IF (.PUSH [PUSH$V_PMT])
; 1247          2163  3           THEN
; 1248          2164  4               BEGIN
; 1249          2165  4               CCB [RAB$B_PSZ] = .PUSH [PUSH$B_PSZ];
; 1250          2166  4               CH$MOVE (.CCB [RAB$B_PSZ], PUSH [PUSH$T_PROMPT], .CCB [RAB$L_PBF]);
; 1251          2167  3               END;
; 1252          2168  3
; 1253          2169  3           CH$MOVE (ISB$K_ISB_LEN, PUSH [PUSH$X_ISB], .CCB - ISB$K_ISB_LEN - LUB$K_LUB_LEN);
; 1254          2170  3   !+
; 1255          2171  3   ! If the LUN has been marked for deallocation (which means that it
; 1256          2172  3   ! has been closed but not deallocated yet because it has I/O in
; 1257          2173  3   ! progress) then clear the statement type field so that all
; 1258          2174  3   ! continued I/O will fail.  The statement type must be set so that
; 1259          2175  3   ! the owning language will get an error when I/O continues.
; 1260          2176  3   !-
; 1261          2177  3
; 1262          2178  4           IF (.CCB [LUB$V_DEALLOC])
; 1263          2179  3           THEN
; 1264          2180  3
; 1265          2181  3               CASE .CCB [LUB$B_LANGUAGE] FROM LUB$K_LANG_MIN TO LUB$K_LANG_MAX OF
; 1266          2182  3                   SET
; 1267          2183  3
; 1268          2184  3                   [LUB$K_LA.,G_FOR] :
; 1269          2185  3                       CCB [ISB$B_STTM_TYPE] = ISB$K_FORSTTYLO - 1;
; 1270          2186  3
; 1271          2187  3                   [LUB$K_LANG_BAS] :
; 1272          2188  3                       CCB [ISB$B_STTM_TYPE] = ISB$K_BASSTTYLO - 1;
; 1273          2189  3
; 1274          2190  3                   [LUB$K_LANG_NONE] :
; 1275          2191  3                       CCB [ISB$B_STTM_TYPE] = 0;
; 1276          2192  3
; 1277          2193  3                 [OUTRANGE] :
; 1278          2194  3                     LIB$STOP (OTS$_FATINTERR);
; 1279          2195  3                 TES;
; 1280          2196  3
; 1281          2197  2         END;
; 1282          2198  2
; 1283          2199  2 !+
; 1284          2200  2 ! We are done with the item from the I/O Active List, free it.
; 1285          2201  2 !-
; 1286          2202  3     BEGIN
; 1287          2203  3
; 1288          2204  3     LOCAL
; 1289          2205  3         FREE_VM_RESULT;
; 1290          2206  3
; 1291          2207  3     FREE_VM_RESULT = LIB$FREE_VM (%REF (PUSH$K_LENGTH), PUSH);
; 1292          2208  3
; 1293          2209  3     IF ( NOT .FREE_VM_RESULT) THEN LIB$STOP (OTS$_FATINTERR);
; 1294          2210  3
; 1295          2211  2     END;
; 1296          2212  2     RETURN;
; 1297          2213  1     END;                                     ! of routine POP_ACTIVE



                                                         03FC 00000 POP_ACTIVE·
                                                                            .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9                  ; 2078
                                        59 00000000G   00  9E 00002         MOVAB   OTS$$L_CUR_LUN, R9                            ;
                                        58 00C00000G   00  9E 00009         MOVAB   LIB$STOP, R8                                  ;
                                        57 00000000G   8F  D0 00010         MOVL    #OTS$_FATINTERR, R7                           ;
                                        5E             08  C2 00017         SUBL2   #8, SP                                        ;
                                   04   AE 00000000'   FF  0F 0001A         REMQUE  @OTS$Q_IO_ACTIVE, PUSH                        ; 2125
                                                       05  1C 00022         BVC     1$                                            ;
                                                       57  DD 00024         PUSHL   R7                                            ;
                                        68             01  FB 00026         CALLS   #1, LIB$STOP                                  ;
                                        56       04    AE  D0 00029 1$:     MOVL    PUSH, R6                                      ; 2130
                                        52       10    A6  32 0002D         CVTWL   16(R6), LUN                                   ;
                          05       14   A6             01  E1 00031         BBC     #1, 20(R6), 2$                                ; 2135
                                        69             52  D0 00036         MOVL    LUN, OTS$$L_CUR_LUN                           ; 2137
                                                       7E  11 00039         BRB     8$                                            ;
                                        53       08    A2  9E 0003B 2$:     MOVAB   8(R2), R3                                     ; 2145
                          05 00000000G  00             53  E0 0003F         BBS     R3, OTS$$V_IOINPROG, 3$                       ;
                                                       57  DD 00047         PUSHL   R7                                            ;
                                        68             01  FB 00049         CALLS   #1, LIB$STOP                                  ;
                                        69             52  D0 0004C 3$:     MOVL    LUN, OTS$$L_CUR_LUN                           ; 2153
                                           00000000G00043  7F 0004F         PUSHAQ  OTS$$AA_LUB_TAB[R3]                           ; 2154
                                        5B             9E  D0 00056         MOVL    @(SP)+, CCB                                   ;
                          50            5B             5B  C3 00059         SUBL3   CCB, CCB, R0                                  ; 2155
                                        5B       58 A04B   9E 0005D         MOVAB   88(R0)[CCB], CCB                              ;
                             00000000G  00             5B  D0 00062         MOVL    CCB, OTS$$A_CUR_LUB                           ; 2156
      FC   AB             01            01       14    A6  F0 00069         INSV    20(R6), #1, #1, -4(CCB)                       ; 2157
                                   08   AB       08    A6  7D 00070         MOVQ    8(R6), 8(CCB)                                 ; 2158
                                   1F   AB       13    A6  90 00075         MOVB    19(R6), 31(CCB)                               ; 2160
                          0F       14   A6             02  E1 0007A         BBC     #2, 20(R6), 4$                                ; 2162
                                   34   AB       12    A6  90 0007F         MOVB    18(R6), 52(CCB)                               ; 2165
                             50      34    AB  9A 00084        MOVZBL  52(CCB), R0                          ; 2166
          30   BB        15  A6            50  28 00088        MOVC3   R0, 21(R6), @48(CCB)                 ;
        FEC0   CB        65  A6    00BC    8F  28 0008E 4$:    MOVC3   #188, 101(R6), -298(CCB)             ; 2169
          1D             FF  AB            04  E1 00097        BBC     #4, -1(CCB), 8$                      ; 2178
          02                 00      D8    AB  8F 0009C        CASEB   -40(CCB), #0, #2                     ; 2181
        0014               000D      0014         000A1 5$:    .WORD   7$-5$,-                              ;
                                                                       6$-5$,-                              ;
                                                                       7$-5$                                ;
                                           57  DD 000A7        PUSHL   R7                                   ; 2194
                             68            01  FB 000A9        CALLS   #1, LIB$STOP                         ;
                                           0B  11 000AC        BRB     8$                                   ;
                       FF71  CB            1A  90 000AE 6$:    MOVB    #26, -143(CCB)                       ; 2188
                                           04  11 000B3        BRB     8$                                   ;
                                   FF71    CB  94 000B5 7$:    CLRB    -143(CCB)                            ; 2191
                                     04    AE  9F 000B9 8$:    PUSHAB  PUSH                                 ; 2207
                         04  AE    0121    8F  3C 000BC        MOVZWL  #289, 4(SP)                          ;
                                     04    AE  9F 000C2        PUSHAB  4(SP)                                ;
                  00000000G  00            02  FB 000C5        CALLS   #2, LIB$FREE_VM                      ;
                             05            50  E8 000CC        BLBS    FREE_VM_RESULT, 9$                   ; 2209
                                           57  DD 000CF        PUSHL   R7                                   ;
                             68            01  FB 000D1        CALLS   #1, LIB$STOP                         ;
                                               04 000D4 9$:    RET                                          ; 2213

; Routine Size:  213 bytes,    Routine Base:  _OTS$CODE + 03FF


; 1298          2214  1
```

```
1 GLOBAL ROUTINE OTS$$POP_CCB                          ! Restore old CCB
1     : JSB_CB_POP NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Restore the I/O system to its state before the call to
1 !       PUSH_CCB.  Clear LUB$V_IO ACTIVE.  If the I/O active list
1 !       is empty, clear OTS$$A_CUR_LUB, otherwise set it to
1 !       its previous value and restore its ISB, etc.
1 !
1 !       If virtual memory for a compiled format is allocated for this
1 !       ISB, it is freed.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       CALL OTS$$POP_CCB ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       CCB
1 !       OTS$$AA_LUB_TAB
1 !       OTS$$Q_IO_ACTIVE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       CCB                     Set to previous LUB/ISB/RAB
1 !       OTS$$Q_IO_ACTIVE        Holds one fewer item
1 !       LUB$V_IO_ACTIVE         Cleared to indicate I/O no longer active,
1 !                               but may be set by the pop from the
1 !                               I/O Active list.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       May call LIB$FREE_VM to free virtual memory.
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         CCB : REF BLOCK [, BYTE];
2
2 !+
2 ! If the LUB has been marked for deallocation (by CLOSE) and there is
2 ! no I/O active, deallocate it.  If there is I/O Active, the
2 ! deallocation must be defered until after all of the I/O has completed
2 ! to insure that the continued I/O will get the "I/O continued to closed
2 ! file" error.
2 !-
2
3     IF (.CCB [LUB$V_DEALLOC] AND ( NOT .CCB [ISB$V_RECURSIVE]))
2     THEN
2         DEALLOCATE ()
```

```
2           ELSE
3               BEGIN
3       !+
3       ! This is no longer the unit with I/O active.
3       !-
3               CCB [LUB$V_IO_ACTIVE] = 0;
3       !+
3       ! See if I/O will continue on this unit.  It will continue if
3       ! ISB$V_RECURSIVE is set, which means that PUSH_CCB was called
3       ! with I/O in progress on this LUN.  We make this test before
3       ! restoring the ISB because we may be restoring to the same
3       ! LUN, and the former I/O may be the top level of I/O for this
3       ! LUN, and if so it will have ISB$V_RECURSIVE clear.
3       !-
3
3               IF ((.OTS$$L_LVL_CTR EQL 0) AND (.CCB [ISB$V_RECURSIVE])) THEN LIB$STOP (OTS$_FATINTERR);
3
4               IF ( NOT .CCB [ISB$V_RECURSIVE])
3               THEN
3
4                   IF (TESTBITCC (OTS$$V_IOINPROG [.CCB [LUB$W_LUN] - LUB$K_ILUN_MIN]))
3                   THEN
3                       LIB$STOP (OTS$_FATINTERR);
3
2               END;
2
2       !+
2       ! Since OTS$$V_IOINPROG may now be clear, our CCB may be deallocated, so
2       ! we cannot touch it again.  For that matter, we may have deallocated
2       ! it ourselves above.
2       !
2       ! If there was previous I/O, restore it.  Otherwise return to the idle
2       ! state.
2       !-
2
3           IF (.OTS$$L_LVL_CTR NEQ 0)
2           THEN
2               POP_ACTIVE ()
2           ELSE
3               BEGIN
3               OTS$$A_CUR_LUB = 0;
3               OTS$$L_CUR_LUN = LUB$K_LUN_MAX + 1;
2               END;
2
2       !+
2       ! Decrement the level counter.  If we are at the top level the level
2       ! counter will go from 0 to -1.
2       !-
2           OTS$$L_LVL_CTR = .OTS$$L_LVL_CTR - 1;
2           RETURN;
1           END;                                        ! of routine OTS$$POP_CCB



0B      FF   AB             04  E1 00000 OTS$$POP_CCB::
```

```
                                                                          BBC     #4, -1(CCB), 1$                              ; 2269
                                       07        97  AB  E8 00005         BLBS    -105(CCB), 1$                                :
                                FE14   CF            00  FB 00009         CALLS   #0, DEALLOCATE                               ; 2271
                                                     3D  11 0000E         BRB     4$                                           :
                                  FC   AB            02  8A 00010 1$:     BICB2   #2, -4(CCB)                                  ; 2277
                                          00000000G  00  D5 00014         TSTL    OTS$$L_LVL_CTR                               ; 2287
                                                     11  12 0001A         BNEQ    2$                                           :
                                       11        97  AB  E9 0001C         BLBC    -105(CCB), 3$                                :
                                          00000000G  8F  DD 00020         PUSHL   #OTS$_FATINTERR                              :
                            00000000G  00            01  FB 00026         CALLS   #1, LIB$STOP                                 :
                                       1C        97  AB  E8 0002D 2$:     BLBS    -105(CCB), 4$                                ; 2289
                                       50        C6  AB  32 00031 3$:     CVTWL   -58(CCB), R0                                 ; 2292
                                       50            08  C0 00035         ADDL2   #8, R0                                       :
                         OD 00000000G  00            50  E4 00038         BBSC    R0, OTS$$V_IOINPROG, 4$                      :
                                          00000000G  8F  DD 00040         PUSHL   #OTS$_FATINTERR                              ; 2294
                            00000000G  00            01  FB 00046         CALLS   #1, LIB$STOP                                 :
                                          00000000G  00  D5 0004D 4$:     TSTL    OTS$$L_LVL_CTR                               ; 2307
                                                     07  13 00053         BEQL    5$                                           :
                                FED1   CF            00  FB 00055         CALLS   #0, POP_ACTIVE                               ; 2309
                                                     0E  11 0005A         BRB     6$                                           :
                                          00000000G  00  D4 0005C 5$:     CLRL    OTS$$A_CUR_LUB                               ; 2312
                            00000000G  00        78  8F  9A 00062         MOVZBL  #120, OTS$$L_CUR_LUN                         ; 2313
                                          00000000G  00  D7 0006A 6$:     DECL    OTS$$L_LVL_CTR                               ; 2320
                                                     05  00 00070         RSB                                                  ; 2322

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


; 1408         2323  1
; 1409         2324  1 END                                                !End of module OTS$$CCB
; 1410         2325  1
; 1411         2326  0 ELUDOM





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




;                                   Library Statistics
:
:                                                  -------- Symbols --------        Pages       Processing
:       File                                       Total    Loaded   Percent       Mapped       Time
:
:  _$255$DUA28:[SYSLIB]STARLET.L32;1                9776        20         0          581         00:00.8
```

```
;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:OTSCCB/OBJ=OBJ$: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