```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

```
**FILE**ID**RMOSTALL                                        K  7




RKRRRRRR    MM      MM    000000      SSSSSSSS  TTTTTTTTTT    AAAAAA    LL          LL
RRRRRRRR    MM      MM    000000      SSSSSSSS  TTTTTTTTTT    AAAAAA    LL          LL
RR      RR  MMMM  MMMM  00      00  SS              TT      AA      AA  LL          LL
RR      RR  MMMM  MMMM  00      00  SS              TT      AA      AA  LL          LL
RR      RR  MM  MM  MM  00    0000  SS              TT      AA      AA  LL          LL
RR      RR  MM  MM  MM  00    0000  SS              TT      AA      AA  LL          LL
RRRRRRRR    MM      MM  00  00  00    SSSSSS        TT      AA      AA  LL          LL
RRRRRRRR    MM      MM  00  00  00    SSSSSS        TT      AA      AA  LL          LL
RR  RR      MM      MM  0000    00          SS      TT      AAAAAAAAAA  LL          LL
RR  RR      MM      MM  0000    00          SS      TT      AAAAAAAAAA  LL          LL
RR    RR    MM      MM  00      00          SS      TT      AA      AA  LL          LL              ....
RR    RR    MM      MM  00      00          SS      TT      AA      AA  LL          LL              ....
RR      RR  MM      MM    000000    SSSSSSSS        TT      AA      AA  LLLLLLLLLL  LLLLLLLLLL      ....
RR      RR  MM      MM    000000    SSSSSSSS        TT      AA      AA  LLLLLLLLLL  LLLLLLLLLL      ....


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
```

Table of contents

```
0000            $BEGIN  RMOSTALL,000,RM$RMS0,<STALL FOR I/O COMPLETION>,<NOWRT,QUAD>
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000    ;++
0000    ; Facility: rms32
0000    ;
0000    ; Abstract:
0000    ;                this module includes the various routines to
0000    ;                handle required i/o stalls and the restarting
0000    ;                of a thread upon i/o completion.
0000    ;
0000    ; Environment:
0000    ;                star processor running starlet exec.
0000    ;
0000    ; Author: l f laverdure,        creation date: 4-FEB-1977
0000    ;
0000    ; Modified By:
0000    ;
0000    ;       V03-014 RAS0269         Ron Schaefer            14-Mar-1984
0000    ;               A little performance boost by re-arranging some code
0000    ;               and branches and some instruction optimization.
0000    ;               Correctly probe arglist before asynch copy and
0000    ;               set -1 addr if not accessible so that RM$EXRMS can
0000    ;               give the user an error.
0000    ;
0000    ;       V03-013 DAS0004         David Solomon           02-Feb-1984
0000    ;               In RM$STALL, don't call RM$LOWER_LOCK unless sharing.
0000    ;
0000    ;       V03-012 KPL0001         Peter Lieberwirth       13-May-1983
0000    ;               Change byte immediate MOV to word immediate to account
0000    ;               for increased size of FAB-related ASB.
0000    ;
0000    ;       V03-011 SHZ0001         Stephen H. Zalewski     13-Apr-1983
0000    ;               If we enter stall via RM$STALL_LOCK, set a flag to prevent
```

```
0000  ;                 us from reenqueuing for the lock after it was granted.
0000  ;
0000  ;         V03-010 DAS0003         David Solomon           21-Feb-1983
0000  ;                 Add entry point RM$CHKAST_ANY the same as RM$BLKFINCHK
0000  ;                 for use by any RMS AST routine (e.g. it doesn't validate
0000  ;                 the ASTPRM).
0000  ;
0000  ;         V03-009 KBT0366         Keith B. Thompson       11-Oct-1982
0000  ;                 Check for stack fit with new asb$w_sktlen field
0000  ;
0000  ;         V03-008 KBT0362         Keith B. Thompson       6-Oct-1982
0000  ;                 asb$b_stksiz is now a word field
0000  ;
0000  ;         V03-007 KBT0360         Keith B. Thompson       6-Oct-1982
0000  ;                 Fix check before calling restore_lock
0000  ;
0000  ;         V03-006 KBT0323         Keith B. Thompson       9-Sep-1982
0000  ;                 Remove all S0 sharing code and add new STALL_LOCK test
0000  ;                 on return from stall
0000  ;
0000  ;         V03-005 JWH0002         Jeffrey W. Horn         07-Sep-1982
0000  ;                 Remove test definition of IMP$V_RUH accidentally
0000  ;                 left in JWH0001.  Also fix bugs in JWH0001 in RUSTALL
0000  ;                 logic by restoring R4, R5 before returning to caller and
0000  ;                 correcting AST handling.
0000  ;
0000  ;         V03-004 KBT0217         Keith B. Thompson       23-Aug-1982
0000  ;                 Reorganize psects
0000  ;
0000  ;         V03-003 JWH0001         Jeffrey W. Horn         5-Aug-1982
0000  ;                 Add logic to not stall if called from within
0000  ;                 the RMS recovery unit handler, but to simply wait
0000  ;                 in exec mode unitil I/O completes.
0000  ;
0000  ;         V03-002 KBT0080         Keith B. Thompson       9-Jul-1982
0000  ;                 Add stall_lock entry point
0000  ;
0000  ;         V03-001 KDM0002         Kathleen D. Morse       28-Jun-1982
0000  ;                 Added $PCBDEF.
0000  ;
0000  ;--
0000  ;
0000
```

```
          0000          .SBTTL  DECLARATIONS
          0000
          0000    ;
          0000    ; Include Files:
          0000    ;
          0000
          0000    ;
          0000    ; Macros:
          0000    ;
          0000
          0000          $SETEFDEF                       ; system service $setef definitions
          0000          $IFBDEF
          0000          $IRBDEF
          0000          $ASBDEF
          0000          $FABDEF
          0000          $RABDEF
          0000          $BDBDEF
          0000          $PIODEF
          0000          $PCBDEF
          0000          $IMPDEF
          0000          $RMSDEF
          00C0
          0000    ;
          0000    ; equated symbols
          0000    ;
          0000
00000020  0000          BKP=IRB$L_BKPBITS*8             ; bit offset to bookkeeping bits
```

```
0000          .SBTTL  RM$STALL - STALL FOR I/O COMPLETION ROUTINE
0000
0000    ;++
0000    ;
0000    ; RM$STALL              stall for I/O completion routine
0000    ; RM$STALL_LOCK         alternate entry point for stall for file lock
0000    ; RM$ENBAST             re-enable ASTs
0000    ;
0000    ; this routine is called whenever a stream must stall for either an i/o
0000    ; completion or for access to the shared file database (or part thereof).
0000    ;
0000    ; this routine first checks if the stalling stream is for a shared file
0000    ; and if so, the shared ifab is released.  next it checks to see whether
0000    ; an asb (asynchronous context block) exists, and if not, the stalling stream
0000    ; is for a fab function, and it allocates an asb, saving its address in the
0000    ; ifab.
0000    ;
0000    ; if this is an asychronous rab operation, copies the argument list into
0000    ; the asb, changes the arglist pointer to point to the saved copy, and sets
0000    ; the status code to rms$_pending.
0000    ;
0000    ; if not an asychronous rab operation, sets the status code to rms$_stall.
0000    ;
0000    ; the routine then saves registers r4 thru r11, the stack along
0000    ; with the return pc, and the stack size in the asb.
0000    ;
0000    ; finally the routine checks for running at exec ast level,
0000    ; and if so, merely returns (i.e., it exits from the ast), otherwise
0000    ; it re-enables asts, sets the status code into r0, and returns to the
0000    ; rms user possibly waiting at user's access mode).
0000    ;
0000    ;       return sequence depends upon following registers not being
0000    ;       destroyed by the return thru the change mode dispatcher to
0000    ;       the rms synchronization code:
0000    ;
0000    ;               r8      structure address
0000    ;               r4      $wait type flag (0=same rab, 1=different rab)
0000    ;               r3      efn to synchronize on
0000    ;
0000    ; Calling sequence:
0000    ;
0000    ;       BSBW    RM$STALL
0000    ;
0000    ; Input Parameters:
0000    ;
0000    ;       r11     impure area address
0000    ;       r10     ifab address if r9 is an irab address
0000    ;       r9      ifab/irab address
0000    ;
0000    ; Implicit Inputs:
0000    ;
0000    ; the contents of the ifab/irab and impure area.
0000    ;
0000    ; Output Parameters:
0000    ;
0000    ;       This routine does not return directly to the caller,
0000    ;       exiting from rms instead. return occurs via the routine
```

```
0000  ;        rm$stallast, which is entered via the ast signaling the
0000  ;        completion of the i/o being awaited by rm$stall. upon
0000  ;        return to the caller his entire context with the exception of
0000  ;        r0 thru r3 and ap is restored.
0000  ;
0000  ; Implicit Outputs:
0000  ;
0000  ;        an asb is allocated, if required, and filled in.
0000  ;        the rms event flag may be cleared.
0000  ;
0000  ; Completion Codes:
0000  ;
0000  ;        if returning to caller of rms, r0 will be set to
0000  ;        either rms$_pending (async) of rms$_stall (sync).
0000  ;        if rms$_stall, this code is intercepted by rms
0000  ;        code running in the caller's mode which awaits the
0000  ;        completion of the rms operation.
0000  ;
0000  ;        if exiting from an ast, r0 is undefined.
0000  ;
0000  ; Side Effects:
0000  ;
0000  ; rms asts are reenabled.
0000  ;
0000  ;--
0000
```

```
                              0000
                              0000  ;++
                              0000  ;  note: the following code is not an entry point into this routine
                              0000  ;
                              0000  ;  must allocate an asb for a stalled fab operation.
                              0000  ;  first check for release of sifab.
                              0000  ;  point r11 to pio segment so that if a free page is requred it will be
                              0000  ;  allocated there.
                              0000  ;--
                              0000
                              0000  STALLAL:
                              0000          $TSTPT   STALLAL
                  5B    DD    0006          PUSHL    R11                             ; save impure area addr
5B    00000000'9F    DE    0008          MOVAL    @#PIO$GW_PIOIMPA,R11            ; point to process i/o segment
          51      5B    D0    000F          MOVL     R11,R1                          ; allocate space in control page
      52     58 8F    9A    0012          MOVZBL   #ASB$C_BLN_FAB/4,R2             ; size required
             FFE7'    30    0016          BSBW     RM$GETBLK                       ; go allocate space (r1=addr)
        5B      8E    D0    0019          MOVL     (SP)+,R11                       ; restore impure area addr
           13 50    E9    001C          BLBC     R0,ERRDME
      08 A1     0D    90    001F          MOVB     #ASB$C_BID,ASB$B_BID(R1)        ; make it a real asb
                              0023
                              0023          ASSUME   ASB$W_STKLEN    EQ      0
                              0023
         0130 8F    B0    0023          MOVW     #<ASB$C_BLN_FAB-ASB$C_BLN_FIX>,-; stuff the size of the
                  61          0027                   (R1)                            ;  save stack area
        52      51    D0    0028          MOVL     R1,R2                           ; copy address to right reg
      14 A9     51    D0    002B          MOVL     R1,IFB$L_ASBADDR(R9)            ; save the asb address
             00B5    31    002F          BRW      SYNCOP                          ; join sync operation code
                              0032
                              0032  ;
                              0032  ; couldn't allocate space for an asb
                              0032  ;
                              0032
                              0032  ERRDME: RMSTBUG FTL$_ASBALLFAIL
                              0039
                              0039  ;
                              0039  ; save arglist for async rab operation (first stall only)
                              0039  ;
                              0039
                              0039  ASYNCOP:
5C    00018009 8F    D0    0039          MOVL     #RMS$_PENDING,AP        ; async status code
   50      18 A9    D0    0040          MOVL     IRB$L_ARGLST(R9),R0     ; restore arglist addr
                              0044                                           ; (note: already probed 1st 2 longwords)
 18 A9    0C A2    DE    0044          MOVAL    ASB$L_ARGLST(R2),IRB$L_ARGLST(R9) ; point at temp arglist
      51      60    9A    0049          MOVZBL   (R0),R1                 ; get arg count
   0C A2      80    7D    004C          MOVQ     (R0)+,ASB$L_ARGLST(R2)  ; save count and FAB/RAB addr
              51    D7    0050          DECL     R1                      ; at most 3 args are of interest
              18    1B    0052          BLEQU    15$                     ; branch if o.k.
                              0054          IFNORD   #4,(R0),20$             ; can't read remainder
   14 A2      80    D0    005A          MOVL     (R0)+,ASB$L_ARGLST+8(R2); copy ERR= addr
              51    D7    005E          DECL     R1                      ; count ERR=
              0A    1B    0060          BLEQU    15$                     ; all there is
                              0062          IFNORD   #4,(R0),25$             ; can't read remainder
   18 A2      60    D0    0068          MOVL     (R0),ASB$L_ARGLST+12(R2); copy SUC= addr
              7E    11    006C  15$:    BRB      CTXSAV
                              006E
   14 A2      01    CE    006E  20$:    MNEGL    #1,ASB$L_ARGLST+8(R2)   ; bad ERR= addr
 18 A2    01    CE  0072  25$:     MNEGL   #1,ASB$L_ARGLST+12(R2)   ; bad SUC= addr
          74    11  0076           BRB     CTXSAV
                    0078
                    0078  RUSTALL:
 18 A9    01    CE  0078           MNEGL   #1,IRB$L_ARGLST(R9)                       ; indicate RU hand stall
                    007C           $CLREF_S #IMP$C_ASYEFN                            ; clear event flag
                    0085           CSB     #PIO$V_INHAST,@#PIO$GW_STATUS             ; check for AST disabled
                    008D           $SETAST_S #1                                      ; enable them if so
                    0096           $WAITFR_S #IMP$C_ASYEFN                           ; wait for event flag
 50    0C A9    3C  009F           MOVZWL  IRB$L_IOS(R9),R0                          ; get status
    54    8E    7D  00A3           MOVQ    (SP)+,R4                                  ; restore R4,R5
          05        00A6           RSB                                               ; go back to thread
                    00A7
```

```
                       00A7
                       00A7  ;++
                       00A7  ;
                       00A7  ;  entry point for stall for file lock
                       00A7  ;
                       00A7  ;  NOTE: This entry point assumes that R10 contains the address of the IFAB.
                       00A7  ;
                       00A7  ;--
                       00A7
                       00A7  RMSSTALL_LOCK::
                       00A7          $TSTPT  STALLLOCK
                       00AD          SSB     #IFB$V_STALL_LOCK,(R10)         ; Do not retake the lock once it is
             19   11   00B1          BRB     STALL
                       00B3
                       00B3  ;++
                       00B3  ;
                       00B3  ;  entry point for this routine
                       00B3  ;
                       00B3  ;--
                       00B3
                       00B3  RMSSTALL::
                       00B3          $TSTPT  STALL
                       00B9
                       00B9  ;
                       00B9  ; If sharing, lower file lock to CR.
                       00B9  ;
                       00B9
       07 08 A9   E8   00B9          BLBS    IFB$B_BID(R9),10$          ; branch if R9 ->IFAB (else R10 ->IFAB)
          78 AA   D5   00BD          TSTL    IFB$L_SFSB_PTR(R10)        ; are we sharing?
             0A   13   00C0          BEQL    STALL                      ; no, skip call to RM$LOWER_LOCK
             05   11   00C2          BRB     20$                        ; yes, lower lock on file
          78 A9   D5   00C4  10$:    TSTL    IFB$L_SFSB_PTR(R9)         ; are we sharing?
             03   13   00C7          BEQL    STALL                      ; no, skip call to RM$LOWER_LOCK
          FF34'   30   00C9  20$:    BSBW    RM$LOWER_LOCK              ; lower file lock to CR
                       00CC
        7E   54   7D   00CC  STALL:  MOVQ    R4,-(SP)                   ; Sav? r4,r5
     A5 6B   06   E0   00CF          BBS     #IMP$V_RUH,(R11),RUSTALL; branch if in RU hand
                       00D3
                       00D3          ASSUME  IFB$L_ASBADDR   EQ      IRB$L_ASBADDR
                       00D3
     52   14 A9   D0   00D3          MOVL    IFB$L_ASBADDR(R9),R2       ; get asb address
             03   12   00D7          BNEQ    10$                        ; continue if we have one
           FF24   31   00D9          BRW     STALLAL                    ; stallal if we don't
                       00DC
                       00DC  ;
                       00DC  ; check for asynchronous rab operation and if so copy arglist into the asb
                       00DC  ;
                       00DC
                       00DC          ASSUME  IMP$W_RMSSTATUS EQ      0
                       00DC
     0C 6B   01   E0   00DC  10$:    BBS     #IMP$V_AST,(R11),CTXSAV ; branch if at ast level
                       00E0
                       00E0          ASSUME  IFB$V_ASYNC     EQ      IRB$V_ASYNC
                       00E0
     03 69   23   E1   00E0          BBC     #IRB$V_ASYNC,(R9),SYNCOP; continue if synch operation
           FF52   31   00E4          BRW     ASYNCOP                    ; branch if async operation
                       00E7
```

```
                            00E7  ;
                            00E7  ; synchronous operation first stall  -  set stall i/o status code
                            00E7  ;
                            00E7
                            00E7 SYNCOP: RMSSUC  STALL,AP
                            00EC
                            00EC  ;
                            00EC  ; save stack size, registers, and stack (including return pc)
                            00EC  ;
                            00EC
 53    14 AB    5E    C3    00EC CTXSAV: SUBL3   SP,IMP$L_SAVED_SP(R11),R3       ; get stack size
                            00F1
                            00F1  ;
                            00F1  ;  verify stack fits into asb
                            00F1  ;
                            00F1
                            00F1         ASSUME  ASB$W_STKLEN    EQ      0
                            00F1
          62    53    B1    00F1         CMPW    R3,(R2)                         ; does stack fit?
                44    1A    00F4         BGTRU   ERRBUG                          ; branch if bad
       02 A2    53    B0    00F6         MOVW    R3,ASB$W_STKSIZ(R2)             ; save the size
          52    1C    C0    00FA         ADDL2   #ASB$L_REGS,R2                  ; get addr of register save area
          82    56    D0    00FD         MOVL    R6,(R2)+                        ; save r6
          82    57    7D    0100         MOVQ    R7,(R2)+                        ; save r7 & r8
                            0103
                            0103  ;
                            0103  ; note: r9 saved as ast parameter
                            0103  ;
                            0103
          82    5A    7D    0103         MOVQ    R10,(R2)+               ; save r10 & r11
       62 6E    53    28    0106         MOVC3   R3,(SP),(R2)            ; copy the stack including
                            010A                                         ; saved R4 & R5
                            010A
                            010A  ;
                            010A  ; set the bit in the IFAB/IRAB which indicates that this RMS thread is
                            010A  ; currently stalled. This bit is cleared within RM$STALLAST, when the
                            010A  ; stalled RMS thread resumes.
                            010A  ;
                            010A
                            010A         ASSUME  IFB$V_RMS_STALL EQ      IRB$V_RMS_STALL
                            010A
                            010A         SSB     #IFB$V_RMS_STALL,(R9)   ; set rms stall bit in IRAB/IFAB
                            010E
                            010E  ;
                            010E  ; if really there (just return)
                            010E  ;
                            010E
    1C 6B       01    E4    010E         BBSC    #IMP$V_AST,(R11),RETURN ; clear at ast level and branch
    0D 69       23    E0    0112         BBS     #IRB$V_ASYNC,(R9),30$   ; branch if asynchronous i/o
                            0116
                            0116         ASSUME  IRB$B_EFN       EQ      IFB$B_EFN
                            0116
    53    0B A9     9A      0116         MOVZBL  IRB$B_EFN(R9),R3        ; set event flag on which to wait
                07    12    011A         BNEQ    30$                     ; branch if non-zero (not rah/wbh)
                            011C
                            011C         ASSUME  IFB$V_ASYNC     EQ      IRB$V_ASYNC
                            011C         ASSUME  IFB$V_ASYNCWAIT EQ      IRB$V_ASYNCWAIT
```

```
                                011C
                     18    88   011C              BISB2   #<1@<IRB$V_ASYNC-BKP>>!<1@<IRB$V_ASYNCWAIT-BKP>>,-
                  04 A9         011E                      IRB$L_BKPBITS(R9)        ; show waiting on async efn
               53    1E    D0   0120              MOVL    #IMP$C_ASYEFN,R3         ; and wait on it
                                0123       30$:
                                0123
                                0123       ;++
                                0123       ;
                                0123       ;  at non-ast level  -  re-enable asts
                                0123       ;  entry here from $wait with:
                                0123       ;
                                0123       ;        ap = status
                                0123       ;        r8 = rab address
                                0123       ;        r4 = $wait type flag
                                0123       ;        r3 = efn
                                0123       ;--
                                0123
                                0123       RM$ENBAST::
    04 00000000'9F   00    E5   0123              BBCC    #PIO$V_INHAST,@#PIO$GW_STATUS,ENBAST ; clear ast inhibit
                                012B
                                012B       ;
                                012B       ; branching if clear
                                012B       ;
                                012B
               50    5C    D0   012B       SETSTS: MOVL    AP,R0                    ; restore status code
                           04   012E       RETURN: RET                              ; exit rms
                                012F       ENBAST: $SETAST_S       #1               ; must re-enable asts
                     F1    11   0138              BRB     SETSTS
                                013A
                                013A       ;
                                013A       ; Not enough space in asb for stack.  The bad stack size is in R3.
                                013A       ;
                                013A
                                013A       ERRBUG:
                                013A              RMSTBUG FTL$_STKTOOBIG
```

```
0141              .SBTTL  RM$STALLAST - AST ENTRY POINT FOR I/O COMPLETE
0141
0141        ;++
0141        ;
0141        ; RM$STALLAST:  AST entry point for I/O complete
0141        ; RM$RAHWBHAST: For read ahead/write behind via ast
0141        ; RM$THREADGO:  With r9 already set (for multi buffering).
0141        ;
0141        ; this routine is entered as a result of an ast delivery for i/o
0141        ; completion.  its function is to restart the associated
0141        ; thread which stalled as a result of calling rm$stall.  the
0141        ; following processing is performed:
0141        ;
0141        ;       1. checks for asts inhibited, and if so disables asts,
0141        ;          redeclares the current ast, sets a flag to cause
0141        ;          asts to be re-enabled, and exits.
0141        ;       2. otherwise, restores r9 (ifab or irab address) from
0141        ;          the ast parameter value, checking for a valid ifab
0141        ;          or irab.
0141        ;       3. the asb address is retrieved and the saved
0141        ;          registers (r4-r11) and stack are restored.
0141        ;       4. the user structure (fab or rab) is reprobed.
0141        ;       5. the indicators imp$l_saved_sp and imp$v_ast are set
0141        ;          appropriately
0141        ;       6. if this is a shared file the file lock
0141        ;          is restored for the stream
0141        ;       7. return is made to the routine that called rm$stall
0141        ;          with nearly full context restored (r0-r3 and ap are
0141        ;          destroyed, secondary user structures must be
0141        ;          reprobed, absolute stack addresses are different)
0141        ;
0141        ; Calling sequence:
0141        ;
0141        ;       entered at rm$stallast via an ast.
0141        ;       alternate entry at rm$rahwbhast for read ahead/write behind via ast
0141        ;       alternate entry at rm$threadgo with r9 already set (for multi buffering).
0141        ;
0141        ; Input Parameters:
0141        ;
0141        ;       astprm  - the ifab or irab address
0141        ;       (for rm$rahwbhast astprm = bdb address)
0141        ;
0141        ; Implicit Inputs:
0141        ;
0141        ;       the contents of the ifab or irab and related structures.
0141        ;
0141        ; Output Parameters:
0141        ;
0141        ;       r4-r11  contents before stall
0141        ;       sp      addr of stack having same contents as before stall
0141        ;       pc      restored to return in line after call to rm$stall
0141        ;       r1-r3,ap destroyed
0141        ;       r0      set to contents of 1st word of i/o status block
0141        ;
0141        ; Implicit Outputs:
0141        ;
0141        ;       imp$v_ast               set
```

```
0141    ;       imp$l_saved_sp          set appropriately for new stack
0141    ;
0141    ; Completion Codes:
0141    ;
0141    ;       none
0141    ;
0141    ; Side Effects:
0141    ;
0141    ;       running at ast level.
0141    ;       secondary user structures require re-probing.
0141    ;       absolute stack addresses different.
0141    ;
0141    ;--
0141
```

```
                         0141
                         0141      ;++
                         0141      ;
                         0141      ; entry here via ast for rah/wbh io completion
                         0141      ;
                         0141      ;--
                         0141
                         0141              .ALIGN  QUAD
                         0148              $ENTRY  RM$RAHWBHAST,^/^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>/
                         014A
         54    04 AC  D0 014A              MOVL    4(AP),R4                ; get bdb addr (astprm)
      07 0A A4    06  E3 014E              BBCS    #BDB$V_AST_DCL,BDB$B_FLGS(R4),10$; set i/o done, branching
                         0153                                              ; if no one waiting
                         0153
      04 AC    24 A4  D0 0153              MOVL    BDB$L_WAIT(R4),4(AP)    ; change astprm to irab
                  20  11 0158              BRB     CHECKAST                ; go join common code to restart
                         015A                                              ; stalled stream
                      04 015A      10$:    RET                             ; dismiss ast
                         015B
                         015B      ;++
                         015B      ;
                         015B      ; entry here via ast for recovery-unit io completion
                         015B      ;
                         015B      ;--
                         015B
                         015B              .ALIGN  QUAD
                         0160              $ENTRY  RM$RUSTALLAST,^/^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>/
                0OB3  30 0162              BSBW    RM$CHKAST               ; check for asts inhibited
                         0165
                         0165      ;
                         0165      ; (note this must be a bsbw and
                         0165      ; must immediately follow the entry mask.)
                         0165      ;
                         0165
                         0165      RUSTALLAST:
                         0165              $SETEF_S #IMP$C_ASYEFN                          ; set event flag
                         016E              SSB     #PIO$V_INHAST,@#PIO$GW_STATUS           ; disable ASTs again
                      04 0176              RET
                         0177
                         0177     ;++
                         0177     ;
                         0177     ; entry here via normal i/o completion ast
                         0177     ;
                         0177     ;--
                         0177
                         0177             .ALIGN  QUAD
                         0178             $ENTRY  RM$STALLAST,^/^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>/
                         017A
                         017A     CHECKAST:
               009B   30 017A             BSBW    RM$CHKAST                 ; check for asts inhibited
                         017D
                         017D     ;
                         017D     ; (note this must be a bsbw and
                         017D     ; must immediately follow the entry mask.)
                         017D     ;
                         017D
                         017D     ;
                         017D     ; See if we are within RU handler, if so, handle in RU ast routine
                         017D     ;
                         017D
  50    01      18 A9 C1 017D             ADDL3   IRB$L_ARGLST(R9),#1,R0  ; RU handler stall?
                   E1 13 0182             BEQL    RUSTALLAST                ; branch if so
                         0184
                         0184     RM$THREADGO::
                         0184                                               ; sets r9 = ifab or irab addr
                         0184             $TSTPT  STALAST
                         018A
                         018A     ;
                         018A     ; clear the bit within the IRAB/IFAB indicating that this thread of RMS is
                         018A     ; stalled, as it no longer is, and allow it to continue
                         018A     ;
                         018A
                         018A             ASSUME  IFB$V_RMS_STALL EQ        IRB$V_RMS_STALL
                         018A
                         018A             CSB     #IFB$V_RMS_STALL,(R9)     ; clear rms stall bit in IRAB/IFAB
                         018E
                         018E             ASSUME  IFB$L_ASBADDR   EQ        IRB$L_ASBADDR
                         018E
        51      14 A9 D0 018E             MOVL    IFB$L_ASBADDR (R9),R1     ; get asb addr
                   65 13 0192             BEQL    ERRASB                    ; error if none
                         0194
                         0194             ASSUME  IFB$V_BUSY      EQ        IRB$V_BUSY
                         0194
        61 69      20 E1 0194             BBC     #IRB$V_BUSY,(R9),ERRASB ; branch if stream not busy
        50      02 A1 3C 0198             MOVZWL  ASB$W_STKSIZ(R1),R0       ; get size of stack
           51      1C C0 019C             ADDL2   #ASB$C_REGS,R1            ; move to register save area
                         019F
           56      81 D0 019F             MOVL    (R1)+,R6                  ; restore r6
           57      81 7D 01A2             MOVQ    (R1)+,R7                  ; restore r7/r8
                         01A5
                         01A5     ;
                         01A5     ; note r9 already restored
                         01A5     ;
                         01A5
           5A      81 7D 01A5             MOVQ    (R1)+,R10                 ; restore r10/r11
        14 AB      5E D0 01A8             MOVL    SP,IMP$L_SAVED_SP(R11)  ; save stack entry value
```

```
         5E    50   C2  01AC          SUBL2   R0,SP                   ; allocate required size
      6E 61    50   28  01AF          MOVC3   R0,(R1),(SP)            ; copy stack including return pc
                        01B3
                        01B3          ASSUME  IMP$W_RMSSTATUS EQ      0
                        01B3
         6B    02   88  01B3          BISB2   #<1@IMP$V_AST>,(R11)    ; set flag for at ast level
                        01B6
                        01B6   ;++
                        01B6   ;
                        01B6   ;  restore the file lock mode
                        01B6   ;
                        01B6   ;--
                        01B6
                        01B6          ASSUME  <IFB$C_BID&1>   EQ      1
                        01B6          ASSUME  <IRB$C_BID&1>   EQ      0
                        01B6          ASSUME  IFB$B_BID       EQ      IRB$B_BID
                        01B6
      0B 08 A9      E9  01B6          BLBC    IFB$B_BID(R9),10$                 ; branch if irab
   13 69    37      E4  01BA          BBSC    #IFB$V_STALL_LOCK,(R9),30$        ; branch if stalled for lock and cle
         78 A9      D5  01BE          TSTL    IFB$L_SFSB_PTR(R9)                ; is the file shared?
            0E      13  01C1          BEQL    30$                               ; branch if not
            09      11  01C3          BRB     20$
   08 6A    37      E4  01C5   10$:   BBSC    #IFB$V_STALL_LOCK,(R10),30$       ; branch if stalled for lock and cle
         78 AA      D5  01C9          TSTL    IFB$L_SFSB_PTR(R10)               ; is the file shared?
            03      13  01CC          BEQL    30$                               ; branch if not
         FE2F'      30  01CE   20$:   BSBW    RM$RESTORE_LOCK                   ; restore previous lock mode
                        01D1
                        01D1   ;+
                        01D1   ;
                        01D1   ; reprobe user structure (user could have deleted it from ast or
                        01D1   ; async operation)
                        01D1   ;
                        01D1   ;-
                        01D1
                        01D1          ASSUME  IFB$B_MODE      EQ      IRB$B_MODE
                        01D1          ASSUME  <IFB$C_BID&1>   EQ      1
                        01D1          ASSUME  <IRB$C_BID&1>   EQ      0
                        01D1          ASSUME  IFB$B_BID       EQ      IRB$B_BID
                        01D1
      0E 08 A9      E8  01D1   30$:   BLBS    IFB$B_BID(R9),CHKFAB    ; branch if ifab
                        01D5
                        01D5   ;
                        01D5   ; irab operation
                        01D5   ;
                        01D5
                        01D5          ASSUME  IFB$B_MODE      EQ      IRB$B_MODE
                        01D5          ASSUME  RAB$C_BLN       LE      FAB$C_BLN
                        01D5
                        01D5          IFNOWRT #RAB$C_BLN,(R8),ERRSTRUCT,IRB$B_MODE(R9)
      01    68      91  01DE          CMPB    RAB$B_BID(R8),#RAB$C_BID          ; it must be a rab
            0E      13  01E1          BEQL    GETBACK                           ; branch if so
                        01E3
                        01E3   ;
                        01E3   ; (it could be a forced disconnect, hence a fab)
                        01E3   ;
                        01E3
                        01E3   ;
                        01E3  ; ifab operation
                        01E3  ;
                        01E3
                        01E3  CHKFAB: IFNOWRT #FAB$C_BLN,(R8),ERRSTRUCT,IFB$B_MODE(R9)
            03   68  91 01EC          CMPB    FAB$B_BID(R8),#FAB$C_BID         ; it must be a fab
                 0F  12 01EF          BNEQ    ERRSTRUCT                        ; branch if ok.
                        01F1
                        01F1  ;+
                        01F1  ;
                        01F1  ;  set r0 to status from i/o status block and return to thread
                        01F1  ;
                        01F1  ;-
                        01F1
                        01F1  GETBACK:
            54   8E  7D 01F1          MOVQ    (SP)+,R4                  ; restore r4 and r5
                        01F4
                        01F4          ASSUME  IRB$L_IOS       EQ      IFB$L_IOS
                        01F4
        50   0C A9   3C 01F4          MOVZWL  IRB$L_IOS(R9),R0          ; pick up i/o completion status
                     05 01F8          RSB                               ; restart thread
                        01F9
                        01F9
                        01F9  ;
                        01F9  ;  handle errors
                        01F9  ;
                        01F9
                        01F9  ;
                        01F9  ; no asb found in ifab/irab or stream not busy
                        01F9  ;
                        01F9
                        01F9  ERRASB:
                        01F9          RMSTBUG FTL$_NOASB
                        0200
                        0200  ;
                        0200  ; the user has been playing funny games with memory
                        0200  ;
                        0200
                        0200  ERRSTRUCT:
      58   14 A9   1C   C1 0200          ADDL3   #ASB$L_REGS,IFB$L_ASBADDR(R9),R8        ; point r8 into asb
                        0205
                        0205          ASSUME  <ASB$C_BLN_FAB - ASB$L_REGS> GE FAB$C_BLN
                        0205          ASSUME  FAB$C_BLN       GE      RAB$C_BLN
                        0205
                 0E  BB 0205          PUSHR   #^M<R1,R2,R3>                    ; save regs clobbered by mov
68   0050 8F   00   6E   00  2C 0207          MOVC5   #0,(SP),#0,#FAB$C_BLN,(R8)       ; clear out fake fab/rab
                 0E  BA 020F          POPR    #^M<R1,R2,R3>                    ; restore regs
                 DE  11 0211          BRB     GETBACK                          ; return to thread
                        0213
```

```
0213          .SBTTL  RM$CHKAST - CHECK FOR ASTS INHIBITED
0213
0213    ;++
0213    ;
0213    ; RM$CHKAST:     Check for ASTs inhibited
0213    ; RM$BLKFINCHK:
0213    ; RM$CHKAST_ANY:
0213    ;
0213    ; This routine checks for asts inhibited, and if so disables
0213    ; asts, redeclares the current ast, clears the flag
0213    ; pio$v_inhast to cause asts to be reenabled when the
0213    ; active non-ast code exits, and exits.
0213    ;
0213    ; If asts are not disabled, sets r9 to the value of the
0213    ; ast parameter and checks that it is a valid ifab of
0213    ; irab address, and returns to the caller.
0213    ;
0213    ; The RM$BLKFINCHK and RM$CHKAST_ANY entry points do not validate the AST
0213    ; parameter.
0213    ;
0213    ; calling sequence
0213    ;
0213    ;       BSBW    RM$CHKAST
0213    ;       BSBW    RM$BLKFINCHK
0213    ;       BSBW    RM$CHKAST_ANY
0213    ;
0213    ;
0213    ; Input Parameters:
0213    ;
0213    ;       ap      ast argument list address
0213    ;
0213    ; Implicit Inputs:
0213    ;
0213    ;       it is assumed that rm$chkast was called via bsbw as
0213    ;       the first instruction of the ast routine.
0213    ;
0213    ; Output Parameters:
0213    ;
0213    ;       If return is made to caller,
0213    ;       R9 = AST parameter, which is
0213    ;               ifab or irab address for RM$CHKAST, or
0213    ;               BLB address for RM$BLKFINCHK entry.
0213    ;
0213    ; Implicit outputs:
0213    ;
0213    ;       may requeue the ast if currently inhibited.
0213    ;
0213    ; Condition Codes:
0213    ;
0213    ;       none.
0213    ;
0213    ; Side Effects:
0213    ;
0213    ;       asts may be disabled.
0213    ;
0213    ;--
0213
```

```
                            0213         .ALIGN  QUAD
                            0218
                            0218 RM$CHKAST::
        59   04 AC     D0   0218         MOVL    4(AP),R9                   ; get ifab/irab address
14 00000000'9F   00    E4   021C         BBSC    #PIO$V_INHAST,@#PIO$GW_STATUS,DSBLAST ; branch if inhibited
                            0224
                            0224 ;
                            0224 ; o.k. to receive ast
                            0224 ; check r9 ifab or irab address for validity
                            0224 ;
                            0224
                            0224         ASSUME  IFB$B_BID       EQ      IRB$B_BID
                            0224
        0A   08 A9     91   0224         CMPB    IRB$B_BID(R9),#IRB$C_BID; is it an irab?
                06     13   0228         BEQL    10$                        ; if so exit
        0B   08 A9     91   022A         CMPB    IFB$B_BID(R9),#IFB$C_BID; if not then it must be an ifab?
                01     12   022E         BNEQ    20$                        ; if not an ifab then we goofed
                       05   0230 10$:    RSB                                ; exit
                            0231
                            0231 20$:    RMSTBUG FTL$_BADASTPRM             ; oop-
                            0238
                            0238 ;
                            0238 ; asts are inhibited
                            0238 ; disable asts and redeclare the current ast
                            0238 ;
                            0238
                            0238 DSBLAST:
                            0238         $TSTPT  ASTDSA
                            023E         $SETAST_S       #0                 ; disable asts
        51   8E   05   C3   0247         SUBL3   #5,(SP)+,R1                ; compute ast address
                            024B
                            024B ;
                            024B ; (return pc - 3-byte bsw
                            024B ; - 2-byte entry mask)
                            024B ;
                            024B
                            024B         $DCLAST_S       ASTADR=(R1),ASTPRM=R9; re-declare the ast
             01 50     E9   0258         BLBC    R0,ERRAST
                       04   025B         RET                                ; and exit
                            025C
                            025C ;
                            025C ; no space to declare an ast
                            025C ;
                            025C
                            025C ERRAST: RMSTBUG FTL$_CANTDOAST
                            0263
                            0263 ;+
                            0263 ; Alternate entries.
                            0263 ;-
                            0263
                            0263         .ALIGN  QUAD
                            0268 RM$BLKFINCHK::                         ; Used in RMORELEAS.
                            0268 RM$CHKAST_ANY::                        ; Note: do not validate ASTPRM.
        59   04 AC     D0   0268         MOVL    4(AP),R9               ; get AST parameter.
C4 00000000'9F   00    E4   026C         BBSC    #PIO$V_INHAST,@#PIO$GW_STATUS,DSBLAST ; branch if inhibited
                       05   0274         RSB                            ; Return to caller if not inhibited.
                            0275
```

```
     0275   824          .END
```

```
$$.PSECT_EP                    = 00000000                  IRB$B_EFN                       = 0000000B
$$ARGS                         = 00000001                  IRB$B_MODE                      = 0000000A
$$RMSTEST                      = 0000001A                  IRB$C_BID                       = 0000000A
$$RMS_PBUGCHK                  = 00000010                  IRB$L_ARGLST                    = 00000018
$$RMS_TBUGCHK                  - 00000008                  IRB$L_ASBADDR                   = 00000014
$$RMS_UMODE                    = 00000004                  IRB$L_BKPBITS                   = 00000004
$$T1                           = 00000000                  IRB$L_IOS                       = 0000000C
ASB$B_BID                      = 00000008                  IRB$V_ASYNC                     = 00000023
ASB$C_BID                      = 0000000D                  IRB$V_ASYNCWAIT                 = 00000024
ASB$C_BLN_FAB                  = 00000160                  IRB$V_BUSY                      = 00000020
ASB$C_BLN_FIX                  = 00000030                  IRB$V_RMS_STALL                 = 0000003A
ASB$L_ARGLST                   = 0000000C                  PIO$A_TRACE                       ********   X    01
ASB$L_REGS                     = 0000001C                  PIO$GW_PIOIMPA                    ********   X    01
ASB$W_STKLEN                   = 00000000                  PIO$GW_STATUS                     ********   X    01
ASB$W_STKSIZ                   = 00000002                  PIO$V_INHAST                    = 00000000
ASYNCOP                          00000039 R      01        RAB$B_BID                       = 00000000
BDB$B_FLGS                     = 0000000A                  RAB$C_BID                       = 00000001
BDB$L_WAIT                     = 00000024                  RAB$C_BLN                       = 00000044
BDB$V_AST_DCL                  = 00000006                  RETURN                            0000012E R      01
BKP                            = 00000020                  RM$BLKFINCHK                      00000268 RG     01
CHECKAST                         0000017A R      01        RM$BUG                            ********   X    01
CHKFAB                           000001E3 R      01        RM$CHKAST                         00000218 RG     01
CTXSAV                           000000EC R      01        RM$CHKAST_ANY                     00000268 RG     01
DSBLAST                          00000238 R      01        RM$ENBAST                         00000123 RG     01
ENBAST                           0000012F R      01        RM$GETBLK                         ********   X    01
ERRASB                           000001F9 R      01        RM$LOWER_LOCK                     ********   X    01
ERRAST                           0000025C R      01        RM$RAHWBAST                       00000148 RG     01
ERRBUG                           0000013A R      01        RM$RESTORE_LOCK                   ********   X    01
ERRDME                           00000032 R      01        RM$RUSTALLAST                     00000160 RG     01
ERRSTRUCT                        00000200 R      01        RM$STALL                          000000B3 RG     01
FAB$B_BID                      = 00000000                  RM$STALLAST                       00000178 RG     01
FAB$C_BID                      = 00000003                  RM$STALL_LOCK                     000000A7 RG     01
FAB$C_BLN                      = 00000050                  RM$THREADGO                       00000184 RG     01
FTL$_ASBALLFAIL                = FFFFFFF9                  RMS$_PENDING                    = 00018009
FTL$_BADASTPRM                 = FFFFFFF8                  RMS$_STALL                      = 00018001
FTL$_CANTDOAST                 = FFFFFFF7                  RUSTALL                           00000078 R      01
FTL$_NOASB                     = FFFFFFF5                  RUSTALLAST                        00000165 R      01
FTL$_STKTOOBIG                 = FFFFFFFE                  SETEF$_EFN                      = 00000004
GETBACK                          000001F1 R      01        SETEF$_NARGS                    = 00000001
IFB$B_BID                      = 00000008                  SETSTS                            0000012B R      01
IFB$B_EFN                      = 0000000B                  STALL                             000000CC R      01
IFB$B_MODE                     = 0000000A                  STALLAL                           00000000 R      01
IFB$C_BID                      = 0000000B                  SYNCOP                            000000E7 R      01
IFB$L_ASBADDR                  = 00000014                  SYS$CLREF                         ********  GX    01
IFB$L_IOS                      = 0000000C                  SYS$DCLAST                        ********  GX    01
IFB$L_SFSB_PTR                 = 00000078                  SYS$SETAST                        ********  GX    01
IFB$V_ASYNC                    = 00000023                  SYS$SETEF                         ********  GX    01
IFB$V_ASYNCWAIT                = 00000024                  SYS$WAITFR                        ********  GX    01
IFB$V_BUSY                     = 00000020                  TPT$L_ASTDSA                      ********   X    01
IFB$V_RMS_STALL                = 0000003A                  TPT$L_STALAST                     ********   X    01
IFB$V_STALL_LOCK               = 00000037                  TPT$L_STALL                       ********   X    01
IMP$C_ASYEFN                   = 0000001E                  TPT$L_STALLAL                     ********   X    01
IMP$L_SAVED_SP                 = 00000014                  TPT$L_STALLLOCK                   ********   X    01
IMP$V_AST                      = 00000001
IMP$V_RUH                      = 00000006
IMP$W_RMSSTATUS                = 00000000
IRB$B_BID                      = 00000008
```

```
                                                  +----------------+
                                                  ! Psect synopsis !
                                                  +----------------+

PSECT name                      Allocation           PSECT No.  Attributes
----------                      ----------           ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (   0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
RMSRMS0                         00000275  (   629.)  01 (   1.)    PIC    USR    CON    REL    GBL NOSHR   EXE   RD   NOWRT NOVEC QUAD
$ABS$                           00000000  (     0.)  02 (   2.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE

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

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                    35     00:00:00.10    00:00:01.26
Command processing               127     00:00:00.73    00:00:05.37
Pass 1                           362     00:00:12.77    00:00:31.10
Symbol table sort                  0     00:00:01.53    00:00:03.37
Pass 2                           148     00:00:02.97    00:00:07.55
Symbol table output               13     00:00:00.12    00:00:00.34
Psect synopsis output              2     00:00:00.03    00:00:00.03
Cross-reference output             0     00:00:00.00    00:00:00.00
Assembler run totals             689     00:00:18.26    00:00:49.03

The working set limit was 1650 pages.
69965 bytes (137 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1219 non-local and 18 local symbols.
824 source lines were read in Pass 1, producing 16 object records in Pass 2.
39 pages of virtual memory were used to define 38 macros.

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

Macro library name                           Macros defined
------------------                           --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                        17
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                         4
_$255$DUA28:[SYSLIB]STARLET.MLB;2                     13
TOTALS (all libraries)                                34

1406 GETS were required to define 34 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:RMOSTALL/OBJ=OBJ$:RMOSTALL MSRC$:RMOSTALL/UPDATE=(ENH$:RMOSTALL)+EXECML$/LIB+LIB$:RMS/LIB
```

0320 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RM0XPFN
LIS

RM0SETDID
LIS

RM0SHARE
LIS

RM0WILD
LIS

RM0XAB
LIS

RM0STALL
LIS