```
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
```

```
_$
Sy
---
NT$
NT$
NT$
NT$
NT$
NT$

NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$

NT$

NT$
NT$
NT$
NT$
NT$
NT$
NT

NT
NT
NT
NT
NT
NT
PI
```

```
**FILE**ID**RMSOEXTEN                                                M 13                                                             RM
                                                                                                                                   VO








          RRRRRRRR    MM      MM    SSSSSSSS    000000    EEEEEEEEEE  XX      XX  TTTTTTTTTT  EEEEEEEEEE  NN      NN
          RRRRRRRR    MM      MM    SSSSSSSS    000000    EEEEEEEEEE  XX      XX  TTTTTTTTTT  EEEEEEEEEE  NN      NN
          RR      RR  MMMM  MMMM  SS          00      00  EE          XX      XX      TT      EE          NN      NN
          RR      RR  MMMM  MMMM  SS          00      00  EE          XX      XX      TT      EE          NN      NN
          RR      RR  MM  MM  MM  SS          00    0000  EE            XX  XX        TT      EE          NNNN    NN
          RR      RR  MM  MM  MM  SS          00    0000  EE            XX  XX        TT      EE          NNNN    NN
          RRRRRRRR    MM      MM    SSSSSS    00  00  00  EEEEEEEE        XX          TT      EEEEEEEE    NN  NN  NN
          RRRRRRRR    MM      MM    SSSSSS    00  00  00  EEEEEEEE        XX          TT      EEEEEEEE    NN  NN  NN
          RR  RR      MM      MM          SS  0000    00  EE            XX  XX        TT      EE          NN    NNNN
          RR  RR      MM      MM          SS  0000    00  EE            XX  XX        TT      EE          NN    NNNN
          RR    RR    MM      MM          SS  00      00  EE          XX      XX      TT      EE          NN      NN      ....
          RR    RR    MM      MM          SS  00      00  EE          XX      XX      TT      EE          NN      NN      ....
          RR      RR  MM      MM  SSSSSSSS      000000    EEEEEEEEEE  XX      XX      TT      EEEEEEEEEE  NN      NN      ....
          RR      RR  MM      MM  SSSSSSSS      000000    EEEEEEEEEE  XX      XX      TT      EEEEEEEEEE  NN      NN      ....


          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          $BEGIN  RMSOEXTEN,000,RM$RMS,<DISPATCH FOR EXTEND OPERATION>
0000
0000    ;
```

```
0000  ;++
0000  ;
0000  ; Facility: rms32
0000  ;
0000  ; Abstract:
0000  ;               this module is the highest level control routine
0000  ;               to perform the $extend function.
0000  ;
0000  ; Environment:
0000  ;               star processor running starlet exec.
0000  ;
0000  ; Author:       L F Laverdure,   creation date: 11-JAN-1978
0000  ;
0000  ; Modified By:
0000  ;
0000  ;       V03-012 RAS0284         Ron Schaefer            29-Mar-1984
0000  ;               Fix error paths to put the area id in the STV.
0000  ;
0000  ;       V03-011 SHZ0001         Stephen H. Zalewski,    13-Mar-1984
0000  ;               When you allocate a BDB, you must bump the AVLCL counter.
0000  ;
0000  ;       V03-010 DAS0001         David Solomon           12-Nov-1983
0000  ;               Fix incorrect register use (R5 instead of R6) in RM$CARVE_BDB.
0000  ;
0000  ;       V03-009 CWH3009         CW Hobbs                29-Oct-1983
0000  ;               Fix two uses of R9 to R10 to correct an accvio on
0000  ;               an ISAM file path.
0000  ;
0000  ;       V03-008 KPL0001         Peter Lieberwirth       27-Oct-1983
0000  ;               To journal extends before a connect is done the
0000  ;               BDB and buffer allocated need to contain extra BDBs
0000  ;               and buffers.
0000  ;
0000  ;       V03-007 KBT0543         Keith B. Thompson       10-Jun-1983
0000  ;               Fix broken branch
0000  ;
0000  ;       V03-006 RAS0140         Ron Schaefer            24-Mar-1983
0000  ;               Fix bugcheck caused by forgotten BLB for relative
0000  ;               and shared sequential files if an error occurs on
0000  ;               the actual extend (diskquota probably).
0000  ;
0000  ;       V03-004 KBT0315         Keith B. Thompson       8-Sep-1982
0000  ;               Remove all of the old SO sharing code
0000  ;
0000  ;       V03-003 KBT0181         Keith B. Thompson       23-Aug-1982
0000  ;               Reorganize psects and rename entry point to single '$'
0000  ;
0000  ;       V03-002 KBT0091         Keith B. Thompson       13-Jul-1982
0000  ;               Clean up psects
0000  ;
0000  ;       V03-001 CDS0003         C Saether       30-Mar-1982
0000  ;               Always allocate buffer if block i/o in
0000  ;               RM$ALLOC_BUF routine.
0000  ;
0000  ;--
0000  ;
0000
```

```
                   0000            .SBTTL  DECLARATIONS
                   0000
                   0000    ;
                   0000    ; Include Files:
                   0000    ;
                   0000
                   0000    ;
                   0000    ; Macros:
                   0000    ;
                   0000
                   0000            $CSHDEF
                   0000            $DEVDEF
                   0000            $FABDEF
                   0000            $IFBDEF
                   0000            $IRBDEF
                   0000            $RLSDEF
                   0000            $XABDEF
                   0000            $XABALLDEF
                   0000            $BDBDEF
                   0000            $RJRDEF
                   0000            $RMSDEF
                   0000
                   0000    ;
                   0000    ; Equated Symbols:
                   0000    ;
                   0000
         00000020  0000            FOP=FAB$L_FOP*8
                   0000
                   0000    ;
                   0000    ; Own Storage:
                   0000    ;
                   0000    ;
                   0000    ;  argument list for xab chain processing (allocation xabs only)
                   0000    ;
                   0000
                   0000    XAB_ARGS:
      00 00'20 14  0000            .BYTE   XAB$C_ALL,XAB$C_ALLLEN,XBC$C_EXTALL,0
```

```
0004            .SBTTL  RMS$EXTEND - $EXTEND ROUTINE
0004
0004    ;++
0004    ;
0004    ; RMS$EXTEND
0004    ;
0004    ;  this routine performs the highest level $extend processing.
0004    ;  its functions include:
0004    ;
0004    ;       1. common setup
0004    ;       2. check for all streams idle, exiting if not
0004    ;       3. perform validity checking
0004    ;       4. call main body of extend logic (extend_it) subroutine - first thing
0004    ;          it must do is to call the co-routine alloc_buf to allocate
0004    ;          buffer and lock bdb if necessary.  it does co-routine call back
0004    ;          if buffers allocated so that it can return them when extend_it
0004    ;          exits, else does rsb (will not return to it) if no action required.
0004    ;       5. lock prolog for rel. f. o. not block i/o accessed
0004    ;       6. process any allocation xabs, performing the indicated extends
0004    ;          and bucket formatting if relative.
0004    ;          isam subroutine in rm3face will be called to do the isam stuff.
0004    ;       7. if no allocation xabs present, perform the extend based upon the fab.
0004    ;       8. release the prolog if locked (relative f.o.)
0004    ;       9. exit from extend_it may be call back to alloc_buf to cleanup
0004    ;          buffers if allocated - this preserves error code in r0.
0004    ;       10. exit to the user, generating an ast if requested
0004    ;
0004    ;
0004    ; Calling sequence:
0004    ;
0004    ;       entered from exec as a result of user's calling sys$extend
0004    ;       (e.g., by using the $extend macro).
0004    ;
0004    ; Input Parameters:
0004    ;
0004    ;       ap      user's argument list addr
0004    ;
0004    ; Implicit Inputs:
0004    ;
0004    ;       the contents of the fab and possible related user interface
0004    ;       blocks.
0004    ;
0004    ; Output Parameters:
0004    ;
0004    ;       r0      status code
0004    ;       r1      destroyed
0004    ;
0004    ; Implicit Outputs:
0004    ;
0004    ;       the size of the extension is returned in fab$l_alq or xab$l_alq
0004    ;       if allocation xab(s) present.
0004    ;
0004    ;       a completion ast is queued if so specified by the user.
0004    ;
0004    ; Completion Codes:
0004    ;
0004    ;       standard rms (see functional spec for list).
```

```
                      0004   180 ;
                      0004   181 ; Side Effects:
                      0004   182 ;
                      0004   183 ;       any locate mode pointer for the file is invalidated.
                      0004   184 ;
                      0004   185 ;--
                      0004   186
                      0004   187 ;++
                      0004   188 ;
                      0004   189 ;  extended branches for error conditions
                      0004   190 ;
                      0004   191 ;--
                      0004   192
       FFF9'  31      0004   193 ERRIOP: BRW     RMS$ERRIOP              ; extend on non-disk device
       FFF6'  31      0007   194 ERRFAC: BRW     RMS$ERRFAC              ; file not write accessed
                      000A   195
                      000A   196 ;++
                      000A   197 ;
                      000A   198 ;  entry point for $extend service
                      000A   199 ;
                      000A   200 ;--
                      000A   201
                      000A   202         $ENTRY  RMS$EXTEND
                      000A   203         $TSTPT  EXTEND
       FFED'  30      0010   204         BSBW    RMS$FSET                ; do common setup
                      0013   205                                         ; note: does not return on error
06 6A    3E   E1      0013   206         BBC     #IFB$V_DAP,(R10),10$    ; Check for network operation
       FFE6'  30      0017   207         BSBW    NT$EXTEND               ; Extend file on remote system
       FFE3'  31      001A   208         BRW     RMS$EXRMS               ;  and exit RMS
                      001D   209
                      001D   210 ;
                      001D   211 ; Check that device is disk and that we are write accessed
                      001D   212 ;
                      001D   213
E3 6A    1C   E1      001D   214 10$:    BBC     #DEV$V_RND,IFB$L_PRIM_DEV(R10),ERRIOP; branch if not disk
E2 6A    30   E1      0021   215         BBC     #IFB$V_WRTACC,(R10),ERRFAC; or if not write accessed
         03   10      0025   216         BSBB    EXTEND_IT               ; call extend routine
       FFD6'  31      0027   217         BRW     RMS$EXRMS               ; and exit rms
```

```
                          002A
                          002A     ;
                          002A     ;  main body of extend code to perform extend function.
                          002A     ;  first call alloc_buf which will allocate buffer and necessary control
                          002A     ;  blocks if no streams are connected.  if a stream is already connected,
                          002A     ;  no action is necessary as buffer and control blocks will already be present.
                          002A     ;
                          002A
                          002A     EXTEND_IT:
         31 22 AA   05  E0  002A             BBS     #IFB$V_BIO,IFB$B_FAC(R10),GETXAB; if block i/o, just go
                          002F                                                   ; direct to allocate space to
                          002F                                                   ; file.  no checks on rel or idx
                          002F
                          002F             ASSUME  IFB$C_SEQ       EQ      0
                          002F
              23 AA   95  002F             TSTB    IFB$B_ORGCASE(R10)      ; sequential file ?
                 2C   13  0032             BEQL    GETXAB                  ; if seq,  skip buffer alloc
         55   5E AA   9A  0034             MOVZBL  IFB$B_BKS(R10), R5      ; bucket size in blocks for alloc_buf
            56   01   D0  0038             MOVL    #1, R6                  ; needs a lock blb.
                 52   10  003B             BSBB    RM$ALLOC_BUF            ; see comments above
              4E 50   E9  003D             BLBC    R0,RETURN               ; out on error
         23 AA   01   91  0040             CMPB    #IFB$C_REL,IFB$B_ORGCASE(R10); relative file ?
                 08   12  0044             BNEQ    10$                     ; branch if isam file
         00000000'EF  16  0046             JSB     RM$LOCK_PROLOG          ; read and lock prolog for rel
                 0B   11  004C             BRB     20$                     ; join rest of code
                 52   D4  004E     10$:    CLRL    R2                      ; zero buffer size
                          0050             $CACHE  VBN=#1,-                ; lock vbn 1 to extend
                          0050                     FLAGS=<LOCK,NOREAD,NOBUFFER>
              32 50   E9  0059     20$:    BLBC    R0,RETURN               ; exit on error
         6C AA   54   D0  005C             MOVL    R4,IFB$L_LOCK_BDB(R10)  ; save bdb address
                          0060
                          0060     ;
                          0060     ;  process allocation xab(s), if any.
                          0060     ;
                          0060     ;  the subroutine rm$extend_xab is called for each allocation xab found.
                          0060     ;
                          0060     ;
                          0060
         5C    9D AF  9E  0060     GETXAB: MOVAB   XAB_ARGS,AP             ; set arg list addr
               FF99'  30  0064             BSBW    RM$XAB_SCAN             ; go look for xab
              0C 50   E9  0067             BLBC    R0,EXIT                 ; branch on error
                 54   D5  006A             TSTL    R4                      ; any xabs found?
                 08   12  006C             BNEQ    EXIT                    ; branch if yes, allocation
                          006E                                             ; occured processing xab(s)
                          006E     ;
                          006E     ;  there were no allocation xabs found.
                          006E     ;  use the alq field of the fab to specify the size of the allocation.
                          006E     ;
                          006E
            53   58   D0  006E             MOVL    R8,R3                   ; cause fab alq to be used
                          0071                                             ; xab, fab inputs at same offset
                 56   D4  0071             CLRL    R6                      ; say no xab
               00B1   30  0073             BSBW    XTND                    ; go do the extend
                          0076
                          0076             ASSUME  IFB$C_SEQ       EQ      0
                          0076
              23 AA   95  0076     EXIT:   TSTB    IFB$B_ORGCASE(R10)      ; is this sequential f.o. ?
```

```
                 0F   13  0079          BEQL    CLRTEF                  ; branch if yes
       0E 22 AA  05   E0  007B          BBS     #IFB$V_BIO,IFB$B_FAC(R10),RETURN; return if block i/o accessed
          54  6C AA   D0  0080          MOVL    IFB$L_LOCK_BDB(R10),R4  ; restore lock bdb address
          00000000'EF 17  0084          JMP     RM$RLSPLG               ; release lock and exit routine
                          008A                                          ; with rsb which will deallocate
                          008A                                          ; buffers if allocated before
                          008A   ;
                          008A   ;  clear the auto extend truncate at eof flag
                          008A   ;
                          008A
                          008A   CLRTEF: CSB    #IFB$V_TEF,(R10)
                          008E
                          008E   ;
                          008E   ;  all set
                          008E   ;
                          008E
                          008E   RETURN:
                      05  008E          RSB                             ; call back to alloc_buf
                          008F                                          ; or return to caller
```

```
                              008F           .SBTTL  RM$ALLOC_BUF - CO-ROUTINE TO ALLOCATE/DEALLOCATE BUFFERS
                              008F
                              008F     ;++
                              008F     ;
                              008F     ; RM$ALLOC_BUF
                              008F     ;
                              008F     ;  this routine is called to allocate a buffer and lock bdb for use in extend
                              008F     ;  processing if no streams connected.  the caller is called back so that
                              008F     ;  control returns to this routine to return the buffer when the caller
                              008F     ;  does an rsb.  if no action was taken on the initial call to this routine,
                              008F     ;  control will not return here.
                              008F     ;
                              008F     ;  inputs:
                              008F     ;
                              008F     ;       r11             impure area address
                              008F     ;       r10             ifab address
                              008F     ;       r5              size of buffer desired in blocks
                              008F     ;       r6              1 if lock blb needed, 0 otherwise.
                              008F     ;
                              008F     ;  outputs:
                              008F     ;
                              008F     ;       r9              set to ifab addr after initial return if buff allocated
                              008F     ;       r1-r6           destroyed
                              008F     ;
                              008F     ;       r0              status of buffer allocation if performed, else success
                              008F     ;                       after initial return
                              008F     ;
                              008F     ;
                              008F     ;--
                              008F
                              008F     RM$ALLOC_BUF::
              50    01    D0  008F             MOVL    #1,R0                   ; assume success
        06 22 AA    05    E0  0092             BBS     #IFB$V_BIO, IFB$B_FAC(R10), 10$ ; always allocate if bio.
              1C AA       D5  0097             TSTL    IFB$L_IRAB_LNK(R10)     ; any streams connected?
                    01    13  009A             BEQL    10$                     ; NEQ if so, then just return and
                          05  009C             RSB                             ; no callback will occur.
          55    55    09  78  009D     10$:    ASHL    #9,R5,R5                ; size of buffer in bytes
                    59    D4  00A1             CLRL    R9                      ; don't want bcnt (no irab)
                              00A3
                              00A3     ;
                              00A3     ; If AI or BI journaling, allocate a bigger buffer.  The bigger buffer
                              00A3     ; contains the buffer (as originally required), a BDB to describe the
                              00A3     ; journal operation, and an RJR to describe the journal entry.
                              00A3     ; (For a nice picture of this, see RM3CONN.)
                              00A3     ;
                              00A3
                    55    DD  00A3             PUSHL   R5                              ; save "real" buffer size
      07 00A0 CA    03    E1  00A5             BBC     #IFB$V_AI,IFB$B_JNLFLG(R10),11$ ; skip if not AI jnling
     55   00000094  8F    C0  00AB             ADDL2   #<RJR$C_BKTLEN+BDB$C_BLN>,R5    ; extra BDB and RJR if AI jnling
      08 00A0 CA    02    E1  00B2     11$:    BBC     #IFB$V_BI,IFB$B_JNLFLG(R10),12$ ; skip if not BI journaling
 55   6E   00000094 8F    C1  00B8             ADDL3   #<RJR$C_BKTLEN+BDB$C_BLN>,(SP),R5 ; add in more for BI
                 FF3D'    30  00C0     12$:    BSBW    RM$BDBALLOC_ALT                 ; get the buffer
                  55    8ED0  00C3             POPL    R5                              ; restore "real" buffer size
              59    5A    D0  00C6             MOVL    R10,R9                          ; restore ifab addr to r9
              0F    50    E9  00C9             BLBC    R0,20$                          ; on error, cleanup and return
            00A0 C9       95  00CC             TSTB    IFB$B_JNLFLG(R9)                ; any journaling?
                    07    13  00D0             BEQL    13$                             ; skip if none
          54   40 A9    D0  00D2                 MOVL    IFB$L_BDB_FLNK(R9),R4          ; get BDB address for CARVE_BDB
               00C8     30  00D6                 BSBW    RM$CARVE_BDB                   ; arrange journaling
                            00D9      13$:
                  9E    16  00D9                 JSB     @(SP)+                  ; call the caller back.  when
                            00DB                                                 ;  caller rsb's, come back here.
                  50    DD  00DB      20$:       PUSHL   R0                      ; Save status.
               FF20'    30  00DD                 BSBW    RM$RELEASALL            ; Return all buffers, desc., unmap
                            00E0                                                 ;  global section if neccessary.
                  50  8ED0  00E0                 POPL    R0                      ; Restore status.
      17 22 AA    05    E1  00E3                 BBC     #IFB$V_BIO,IFB$B_FAC(R10),40$ ; All done if not block i/o.
            55    5A    D0  00E8                 MOVL    R10, R5                 ; Get ifab addr into R5.
                            00EB
                            00EB                 ASSUME  IFB$L_IRAB_LNK  EQ      IRB$L_IRAB_LNK
                            00EB
          55   1C A5    D0  00EB      30$:       MOVL    IFB$L_IRAB_LNK(R5), R5  ; any more streams connected?
                  0E    13  00EF                 BEQL    40$                     ; EQL no streams, just return.
                  50    DD  00F1                 PUSHL   R0                      ; Save status code.
               FF0A'    30  00F3                 BSBW    RM$ALBDB                ; Restore BDB for block i/o.
            0084 CA     B6  00F6                 INCW    IFB$W_AVLCL(R10)        ; Bump local buffer count.
                  50  8ED0  00FA                 POPL    R0                      ; Restore status code
                  EC    11  00FD                 BRB     30$                     ; Look for more.
                        05  00FF      40$:       RSB                             , And return.
                        0100          .SBTTL  RM$EXTEND_XAB - ROUTINE TO HANDLE ALLOCATION XABS
                        0100
                        0100    ;++
                        0100    ;
                        0100    ; RM$EXTEND_XAB
                        0100    ;
                        0100    ;  this routine is called from rm$xab_scan whenever an allocation xab
                        0100    ;  is found on an $extend.
                        0100    ;
                        0100    ;  the xab is checked for validity and its parameters are used to extend
                        0100    ;  the file.  if there are no errors, this routine returns to continue
                        0100    ;  the xab scan.
                        0100    ;
                        0100    ;  inputs:
                        0100    ;
                        0100    ;       @sp      return address if successful
                        0100    ;       ap       rm$xab_scan argument list addr
                        0100    ;       r11      impure area addr
                        0100    ;       r10      ifab addr
                        0100    ;       r9       ifab addr
                        0100    ;       r8       fab addr
                        0100    ;       r4       zero if this is the first call, else the area i.d. of
                        0100    ;                the last area + 1
                        0100    ;       r3       xab addr
                        0100    ;
                        0100    ;  the xab has already been checked for basic validity
                        0100    ;
                        0100    ;  outputs:
                        0100    ;
                        0100    ;       r4               this xab's area i.d. + 1
                        0100    ;       r0-r2,r5-r6      destroyed
                        0100    ;
                        0100    ;  if an error occurs, the return address is popped from the stack,
                        0100    ;  r0 is set to the error code, and an rsb is performed back to caller
                        0100    ;  of rm$xab_scan.
                        0100    ;
                        0100    ;--
                        0100
                        0100    RM$EXTEND_XAB::
23 AA   02    91        0100          CMPB    #IFB$C_IDX,IFB$B_ORGCASE(R10); if index file
        06    12        0104          BNEQ    3$
00 54   00    E2        0106          BBSS    #0,R4,2$
        09    11        010A    2$:   BRB     5$
11 54   00    E2        010C    3$:   BBSS    #0,R4,ERRIMX             ; branch if duplicate xab
                        0110
                        0110    ;
                        0110    ; !!! note - different test for
                        0110    ;     indexed f.o. required !!!
                        0110    ;
                        0110
   17 A3      95        0110          TSTB    XAB$B_AID(R3)            ; must be area 0
        0F    12        0113          BNEQ    ERRAID                   ; branch if not
   56   53    D0        0115    5$:   MOVL    R3,R6                    ; xab addr to right reg
        0D    10        0118          BSBB    XTND                     ; go perform extension
   03 50      E8        011A          BLBS    R0,10$                   ; branch on success
        51  8ED0        011D          POPL    R1                       ; pop return pc
              05        0120    10$:  RSB
```

```
                    0121   433
                    0121   434 ;++
                    0121   435 ;
                    0121   436 ;  handle errors
                    0121   437 ;
                    0121   438 ;--
                    0121   439
        FEDC'  31   0121   440 ERRIMX: BRW     RMS$ERRIMX                ; duplicate alloc. xab
                    0124   441
        FED9'  31   0124   442 ERRAID: BRW     RMS$ERRAID                ; bad area i.d.
```

```
                             0127
                             0127  ;++
                             0127  ;
                             0127  ;  xtnd subroutine to perform the extend.
                             0127  ;
                             0127  ;  inputs:
                             0127  ;
                             0127  ;       r11      impure area addr
                             0127  ;       r10      ifab addr
                             0127  ;       r9       ifab addr
                             0127  ;       r8       fab addr
                             0127  ;       r6       xab addr if any, else 0
                             0127  ;       r3       xab or fab addr
                             0127  ;
                             0127  ;  outputs:
                             0127  ;
                             0127  ;       r0               status code
                             0127  ;       xab$l_alq or
                             0127  ;       fab$l_alq        # of blocks extended, 0 if none
                             0127  ;       fab$l_stv        fab$l_stv + # of blocks extended
                             0127  ;       r1,r2,r5,r6      destroyed
                             0127  ;
                             0127  ;--
                             0127
           1018 8F     BB    0127  XTND:   PUSHR   #^M<R3,R4,AP>
                             012B
                             012B          ASSUME  XAB$L_ALQ       EQ      FAB$L_ALQ
                             012B
      55    10 A3      D0    012B          MOVL    XAB$L_ALQ(R3),R5        ; set extend size
               69      15    012F          BLEQ    ERRALQ                  ; branch if bad
            10 A3      D4    0131          CLRL    XAB$L_ALQ(R3)           ; and initialize actual extend size
 0E 22 AA      05      E0    0134          BBS     #IFB$V_BIO,IFB$B_FAC(R10),10$
    23 AA      02      91    0139          CMPB    #IFB$C_IDX,IFB$B_ORGCASE(R10)
               08      12    013D          BNEQ    10$
     00000000'EF       16    013F          JSB     RM$EXTEND3
               4E      11    0145          BRB     XTNDXIT
            FEB6'      30    0147  10$:    BSBW    RM$EXTEND0              ; do the extend
            48 50      E9    014A          BLBC    R0,XTNDXIT              ; branch on failure
                             014D
                             014D  ;
                             014D  ; note: r1 = start vbn of extent
                             014D  ;       r6 = end vbn of extent + 1
                             014D  ;
                             014D
      53    6E         D0    014D          MOVL    (SP),R3                 ; restore xab/fab address
 10 A3   56    51      C3    0150          SUBL3   R1,R6,FAB$L_ALQ(R3)     ; calculate extend size
  0C A8   10 A3        C0    0155          ADDL2   FAB$L_ALQ(R3),FAB$L_STV(R8); and add it in to stv for total
 31 22 AA      05      E0    015A          BBS     #IFB$V_BIO,IFB$B_FAC(R10),SEQEXT; branch if block i/o accessed
                             015F          CASE    TYPE=B,SRC=IFB$B_ORGCASE(R10),-
                             015F          DISPLIST=<SEQEXT,RELEXT>        ; dispatch based on file org
               26      11    0168          BRB     SEQEXT                  ; treat like sequential
```

```
                          016A
                          016A  ;++
                          016A  ;
                          016A  ;  relative file extend  -  format the buckets
                          016A  ;
                          016A  ;--
                          016A
                          016A  RELEXT:
      11 6A   38   E0     016A          BBS     #IFB$V_SEQFIL,(R10),SEQSHR      ; branch if seq file shr'd
   00000000'EF   16       016E          JSB     RM$FMT_BKT2                     ; go format the buckets
   00000000'EF   16       0174          JSB     RM$UPD_PROLOG2                  ; and update the prolog
         6C AA   D4       017A  X:      CLRL    IFB$L_LOCK_BDB(R10)             ; say lock bdb gone
            16   11       017D          BRB     XTNDXIT
                          017F  SEQSHR:
   54    6C AA   D0       017F          MOVL    IFB$L_LOCK_BDB(R10),R4          ; set up r4 to release lock bdb
            50   DD       0183          PUSHL   R0                              ; save status
   00000000'EF   16       0185          JSB     RM$SETHEBK                      ; set hbk and release lock on -1
            50 8ED0       018B          POPL    R0                              ; restore status
            EA   11       018E          BRB     X                               ; and exit
                          0190
                          0190  ;++
                          0190  ;
                          0190  ;  sequential file or block i/o extend  -  update high block
                          0190  ;
                          0190  ;--
                          0190
70 AA    56    01   C3    0190  SEQEXT: SUBL3   #1,R6,IFB$L_HBK(R10)            ; set new high block
                          0195
                          0195  XTNDXIT:
        1018 8F   BA      0195          POPR    #^M<R3,R4,AP>                   ; restore regs
                 05       0199          RSB
                          019A
                          019A  ;++
                          019A  ;
                          019A  ;  handle invalid alq value error
                          019A  ;
                          019A  ;--
                          019A
                          019A  ERRALQ:
                          019A          RMSERR  ALQ
            F4   11       019F          BRB     XTNDXIT
```

```
                                 01A1           .SUBTITLE RM$CARVE_BDB
                                 01A1      ;++
                                 01A1      ; RM$CARVE_BDB
                                 01A1      ;
                                 01A1      ; Carve up the bigger buffer into a BDB, RJR, and the original buffer.
                                 01A1      ;
                                 01A1      ; Inputs:
                                 01A1      ;
                                 01A1      ;        R4      points to real BDB.
                                 01A1      ;        R5      has the "real" buffer size.
                                 01A1      ;        R9      has the IFAB address.
                                 01A1      ;
                                 01A1      ; Outputs:
                                 01A1      ;
                                 01A1      ;        BDBs inited.
                                 01A1      ;
                                 01A1      ;--
                                 01A1
                                 01A1      RM$CARVE_BDB:
                                 01A1
                                 01A1      ; R6 will point to the BDB used for the AI journal entry.
                                 01A1
               56   18 A4    D0  01A1           MOVL    BDB$L_ADDR(R4),R6                   ; get buffer address
          29 00A0 C9    03    E1  01A5           BBC     #IFB$V_AI,IFB$B_JNLFLG(R9),10$      ; skip if no AI jnling
            34 A4    56    D0  01AB           MOVL    R6,BDB$L_AI_BDB(R4)                 ; put AI_BDB address in real BDB
                                 01AF
                                 01AF      ;
                                 01AF      ; Initialize the AI_BDB
                                 01AF      ;
                                 01AF
            08 A6    0C    90  01AF           MOVB    #BDB$C_BID,BDB$B_BID(R6)            ; block ID
            09 A6    14    90  01B3           MOVB    #<BDB$C_BLN/4>,BDB$B_BLN(R6)        ; and block length
               66    56    D0  01B7           MOVL    R6,BDB$L_FLINK(R6)                  ; bdb queue is null
            04 A6    56    D0  01BA           MOVL    R6,BDB$L_BLINK(R6)                  ; ...
      16 A6    55   0044 8F    A1  01BE           ADDW3   #RJR$C_BKTLEN,R5,BDB$W_SIZE(R6)     ; size = RJR + "real" buffer
         18 A6    50 A6    9E  01C5           MOVAB   BDB$C_BLN(R6),BDB$L_ADDR(R6)        ; buffer address
                                 01CA
                                 01CA      ;
                                 01CA      ; Now, correct the "real" BDB's buffer address to point past AI_BDB and RJR.
                                 01CA      ;
                                 01CA
18 A4    18 A6    00000044 8F    C1  01CA           ADDL3   #RJR$C_BKTLEN,BDB$L_ADDR(R6),BDB$L_ADDR(R4)
                                 01D4      10$:
          24 00A0 C9    02    E1  01D4           BBC     #IFB$V_BI,IFB$B_JNLFLG(R9),20$   ; skip if not BI jnling
                                 01DA
                                 01DA      ;
                                 01DA      ; Carve out and initialize the BI_BDB.
                                 01DA      ;
                                 01DA      ;        First, calculate BI_BDB address.  The BI_BDB is allocated after the
                                 01DA      ;        actual buffer.  The actual buffer is pointed to by BDB$L_ADDR(R4).
                                 01DA      ;        The actual buffer size is in R5.
                                 01DA      ;
                                 01DA      ; R6 will point to the BDB used for the BI journal entry.
                                 01DA
          56    18 A4    55    C1  01DA           ADDL3   R5,BDB$L_ADDR(R4),R6                ; pointer to BI_BDB
            30 A4    56    D0  01DF           MOVL    R6,BDB$L_BI_BDB(R4)                 ; filled in "real" BDB
            08 A6    0C    90  01E3           MOVB    #BDB$C_BID,BDB$B_BID(R6)            ; block ID
```

```
                09 A6    14   90  01E7   595         MOVB    #<BDB$C_BLN/4>,BDB$B_BLN(R6)      ; and block length
                   66    56   D0  01EB   596         MOVL    R6,BDB$L_FLINK(R6)                ; bdb queue is null
                04 A6    56   D0  01EE   597         MOVL    R6,BDB$L_BLINK(R6)                ; ...
        16 A6   55   0044 8F  A1  01F2   598         ADDW3   #RJR$C_BKTLEN,R5,BDB$W_SIZE(R6)   ; buffer size = RJR + buffer
             18 A6    50 A6   9E  01F9   599         MOVAB   BDB$C_BLN(R6),BDB$L_ADDR(R6)      ; buffer address
                                  01FE   600
                              05  01FE   601 20$:    RSB
                                  01FF   602
                                  01FF   603         .END
```

```
                                                       D 15
RMS0EXTEN                 DISPATCH FOR EXTEND OPERATION          16-SEP-1984 01:18:09  VAX/VMS Macro V04-00        Page  16    RM
Symbol table                                                      5-SEP-1984 16:24:55  [RMS.SRC]RMS0EXTEN.MAR;1          (13)   VO

$$.PSECT_EP     = 00000000          RETURN            0000008E R  01
$$.TMP          = 0000000D          RJR$C_BKTLEN    = 00000044
$$RMSTEST       = 0000001A          RM$ALBDB          ******** X  01
$$RMS_PBUGCHK   = 00000010          RM$ALLOC_BUF      0000008F RG 01
$$RMS_TBUGCHK   = 00000008          RM$BDBALLOC_ALT   ******** X  01
$$RMS_UMODE     = 00000004          RM$CACHE          ******** X  01
BDB$B_BID       = 00000008          RM$CARVE_BDB      000001A1 R  01
BDB$B_BLN       = 00000009          RM$ERRAID         ******** X  01
BDB$C_BID       = 0000000C          RM$ERRFAC         ******** X  01
BDB$C_BLN       = 00000050          RM$ERRIMX         ******** X  01
BDB$L_ADDR      = 00000018          RM$ERRIOP         ******** X  01
BDB$L_AI_BDB    = 00000034          RM$EXRMS          ******** X  01
BDB$L_BI_BDB    = 00000030          RM$EXTEND0        ******** X  01
BDB$L_BLINK     = 00000004          RM$EXTEND3        ******** X  01
BDB$L_FLINK     = 00000000          RM$EXTEND_XAB     00000100 RG 01
BDB$W_SIZE      = 00000016          RM$FMT_BKT2       ******** X  01
CLRTEF            0000008A R  01    RM$FSET           ******** X  01
CSH$M_LOCK      = 00000001          RM$LOCK_PROLOG    ******** X  01
CSH$M_NOBUFFER  = 00000008          RM$RELEASALL      ******** X  01
CSH$M_NOREAD    = 00000004          RM$RLSPLG         ******** X  01
DEV$V_RND       = 0000001C          RM$SETHEBK        ******** X  01
ERRAID            00000124 R  01    RM$UPD_PROLOG2    ******** X  01
ERRALQ            0000019A R  01    RM$XAB_SCAN       ******** X  01
ERRFAC            00000007 R  01    RMS$EXTEND      = 00000008 RG 01
ERRIMX            00000121 R  01    RMS$_ALQ        = 00018404
ERRIOP            00000004 R  01    SEQEXT            00000190 R  01
EXIT              00000076 R  01    SEQSHR            0000017F R  01
EXTEND_IT         0000002A R  01    TPT$L_EXTEND      ******** X  01
FAB$L_ALQ       = 00000010          X                 0000017A R  01
FAB$L_FOP       = 00000004          XAB$B_AID       = 00000017
FAB$L_STV       = 0000000C          XAB$C_ALL       = 00000014
FOP             = 00000020          XAB$C_ALLLEN    = 00000020
GETXAB            00000060 R  01    XAB$L_ALQ       = 00000010
IFB$B_BKS       = 0000005E          XAB_ARGS          00000000 R  01
IFB$B_FAC       = 00000022          XBC$C_EXTALL      ******** X  01
IFB$B_JNLFLG    = 000000A0          XTND              00000127 R  01
IFB$B_ORGCASE   = 00000023          XTNDXIT           00000195 R  01
IFB$C_IDX       = 00000002
IFB$C_REL       = 00000001
IFB$C_SEQ       = 00000000
IFB$L_BDB_FLNK  = 00000040
IFB$L_HBK       = 00000070
IFB$L_IRAB_LNK  = 0000001C
IFB$L_LOCK_BDB  = 0000006C
IFB$L_PRIM_DEV  = 00000000
IFB$V_AI        = 00000003
IFB$V_BI        = 00000002
IFB$V_BIO       = 00000005
IFB$V_DAP       = 0000003E
IFB$V_SEQFIL    = 00000038
IFB$V_TEF       = 00000036
IFB$V_WRTACC    = 00000030
IFB$W_AVLCL     = 00000084
IRB$L_IRAB_LNK  = 0000001C
NT$EXTEND         ******** X  01
PIO$A_TRACE       ******** X  01
RELEXT            0000016A R  01
```

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

PSECT name                       Allocation           PSECT No.  Attributes
----------                       ----------           ---------  ----------
.  ABS  .                        00000000  (     0.)  00 (   0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
RMSRMS                           000001FF  (   511.)  01 (   1.)    PIC   USR   CON   REL   GBL NOSHR   EXE   RD   NOWRT NOVEC BYTE
$ABS$                            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                        39    00:00:00.09    00:00:00.44
Command processing                   148    00:00:00.72    00:00:05.40
Pass 1                               334    00:00:11.54    00:00:29.33
Symbol table sort                      0    00:00:01.53    00:00:01.87
Pass 2                               112    00:00:02.57    00:00:05.86
Symbol table output                   11    00:00:00.10    00:00:00.21
Psect synopsis output                  3    00:00:00.03    00:00:00.03
Cross-reference output                 0    00:00:00.00    00:00:00.00
Assembler run totals                 649    00:00:16.60    00:00:43.18

The working set limit was 1650 pages.
63802 bytes (125 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1176 non-local and 20 local symbols.
603 source lines were read in Pass 1, producing 14 object records in Pass 2.
29 pages of virtual memory were used to define 28 macros.

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

Macro library name                               Macros defined
------------------                               --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                              18
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                               1
_$255$DUA28:[SYSLIB]STARLET.MLB;2                            5
TOTALS (all libraries)                                      24

1323 GETS were required to define 24 macros.

There were no errors, warnings or information messages.

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

0329 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RMS0FSCN
LIS

RM3XALLO
LIS

RMS0CONN
LIS

RMS0ENTER
LIS

RMS0ERASE
LIS

RMS0CREAT
LIS

RMS0DELET
LIS

RMS0FIND
LIS

RM3XSUMO
LIS

RMS0DISPL
LIS

RM3XKEYO
LIS

RMS0GET
LIS

RMS0CLOSE
LIS

RMS0DISC
LIS

RMS0EXTEN
LIS

RMS0BLKIO
LIS