```
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
```

```
RRRRRRRR    MM      MM      11      RRRRRRRR    EEEEEEEEEE  LL          BBBBBBBB    LL          KK      KK
RRRRRRRR    MM      MM      11      RRRRRRRR    EEEEEEEEEE  LL          BBBBBBBB    LL          KK      KK
RR      RR  MMMM  MMMM    1111      RR      RR  EE          LL          BB      BB  LL          KK      KK
RR      RR  MMMM  MMMM    1111      RR      RR  EE          LL          BB      BB  LL          KK      KK
RR      RR  MM  MM  MM      11      RR      RR  EE          LL          BB      BB  LL          KK    KK
RR      RR  MM  MM  MM      11      RR      RR  EE          LL          BB      BB  LL          KK    KK
RRRRRRRR    MM      MM      11      RRRRRRRR    EEEEEEEE    LL          BBBBBBBB    LL          KKKKKK
RRRRRRRR    MM      MM      11      RRRRRRRR    EEEEEEEE    LL          BBBBBBBB    LL          KKKKKK
RR  RR      MM      MM      11      RR  RR      EE          LL          BB      BB  LL          KK    KK
RR  RR      MM      MM      11      RR  RR      EE          LL          BB      BB  LL          KK    KK
RR    RR    MM      MM      11      RR    RR    EE          LL          BB      BB  LL          KK      KK      ....
RR    RR    MM      MM      11      RR    RR    EE          LL          BB      BB  LL          KK      KK      ....
RR      RR  MM      MM    111111    RR      RR  EEEEEEEEEE  LLLLLLLLLL  BBBBBBBB    LLLLLLLLLL  KK      KK      ....
RR      RR  MM      MM    111111    RR      RR  EEEEEEEEEE  LLLLLLLLLL  BBBBBBBB    LLLLLLLLLL  KK      KK      ....

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
```

```
0000              $BEGIN  RM1RELBLK,000,RM$RMS1,<RELEASE BUFFER FOR SEQ. ORG.>
0000
0000      ;
```

```
0000    ;++
0000    ; Facility: rms32
0000    ;
0000    ; Abstract:
0000    ;               this module releases a buffer causing its
0000    ;               contents to be written out if dirty.
0000    ;
0000    ; Environment:
0000    ;               star processor running starlet exec.
0000    ;
0000    ; Author: L F Laverdure,        creation date: 29-march-77
0000    ;
0000    ; Modified By:
0000    ;
0000    ;       V03-005 DGB0018         Donald G. Blair         02-Mar-1984
0000    ;               Use full-length fib to support access mode protected
0000    ;               files.
0000    ;
0000    ;       V03-004 RAS0179         Ron Schaefer            29-Jul-1983
0000    ;               Change default autoextend size to be 2*MBC now that
0000    ;               we are defaulting MBC to a very large value.
0000    ;               Re  ve use of volume default and clean-up retry logic.
0000    ;
0000    ;       V03-003 KBT0417         Keith B. Thompson       30-Nov-1982
0000    ;               Change ifb$w_devbufsiz to ifb$l_devbufsiz
0000    ;
0000    ;       V03-002 RAS0095         Ron Schaefer             7-Sep-1982
0000    ;               Correct auto-extend logic for overdraft.  Repeal
0000    ;               old algorithm that reported overdraft errors on the
0000    ;               actually entry into overdraft as this lost allocated
0000    ;               blocks.
0000    ;
0000    ;       V03-001 KBT0147         Keith B. Thompson       20-Aug-1982
0000    ;               Reorganize psects
0000    ;
0000    ;       V02-020 JWH0002         Jeffrey W. Horn         12-Jan-1982
0000    ;               Allow retry of auto-extend only on quota-exceeded and
0000    ;               device-full errors from ACP.
0000    ;
0000    ;       V02-019 JWH0001         Jeffrey W. Horn         03-NOV-1981
0000    ;               Changes to use new RMS default extend sizes.
0000    ;
0000    ;       V02-018 REFORMAT        P S Knibbe      25-Jul-1980
0000    ;
0000    ;       V017    PSK0009         P S Knibbe      10-JAN-1980     1:00
0000    ;               changes to accomadate disk overdrafting. don't permit auto-extend
0000    ;               if in overdraft. return error when first go into overdraft.
0000    ;
0000    ; Revision History
0000    ;
0000    ;       E H Marison,    7-JUN-1978  13:06
0000    ;                       add mbf logic and comments
0000    ; 01    -
0000    ;--
0000
```

```
0000          .SBTTL  DECLARATIONS
0000
0000    ;
0000    ; Include Files:
0000    ;
0000
0000    ;
0000    ; Macros:
0000    ;
0000
0000          $SSDEF
0000          $IFBDEF
0000          $IRBDEF
0000          $DEVDEF
0000          $BDBDEF
0000          $RLSDEF
0000          $FIBDEF
0000          $RMSDEF
0000
0000    ;
0000    ; Equated Symbols:
0000    ;
0000
0000    ;
0000    ; Own Storage:
0000    ;
0000
```

```
0000
0000    ;++
0000    ; notes on the multi-block buffering scheme
0000    ;
0000    ; this routine causes multiple blocks to be read together
0000    ; (as specified by mbc) but returned one at a time for
0000    ; processing by the calling routines.
0000    ;
0000    ; assumptions:
0000    ;
0000    ;       1.  mbc is never > 0 except for disk(= # vbn's - 1)
0000    ;       2.  records are always written at eof (only updates
0000    ;           may occur elsewhere in the file).
0000    ;       3.  all sequential i/o calls go thru one of the
0000    ;           following routines:
0000    ;               rm$nxtblk1
0000    ;               rm$wtlst1
0000    ;               rm$relblk1
0000    ;       4.  there is no write sharing for sequential files.
0000    ;       5.  a direct release will be done only when there
0000    ;           is no i/o for the buffer.
0000    ;
0000    ; bdb field usage:
0000    ;
0000    ;       1.  bdb$l_vbn = vbn of first block in buffer
0000    ;       2.  (irb$l_rp_vbn = vbn of current block)
0000    ;       3.  bdb$b_rel_vbn = current vbn rel to start vbn for buffer
0000    ;       4.  bdb$b_val_vbns = # of valid vbns in buffer
0000    ;       5.  bdb$b_flgs:
0000    ;               -bdb$v_drt:all blocks up to the greater of the current vbn
0000    ;                and the number of val_vbns are dirty
0000    ;               -bdb$v_val:the current vbn is valid
0000    ;
0000    ;       6.  the relative vbn = requested vbn - start vbn
0000    ;       7.  current block buffer addr = buff addr + (rel_vbn*512)
0000    ;       8.  bdb$w_numb = # bytes in current block
0000    ;               on reads = (irb$b_mbc+1)*512
0000    ;               or writes = (max(val_vbn,rel_vbn+1))*512
0000    ;       9.  requested vbn is in buffer if its rel_vbn < or = mbc
0000    ;       10. if read required and rel_vbn < val_vbns ok,
0000    ;               else release buffer and reread
0000    ;       11. on release (rm$relblk1) if bdb$v_val is off and the
0000    ;           bdb$v_drt bit is set, merely decrement the
0000    ;           current vbn and set the valid bit.
0000    ;--
0000
```

```
0000             .SBTTL  RM$RELBLK1 - ROUTINE TO RELEASE BLOCK, REWRITING IF DIRTY
0000
0000      ;++
0000      ; RM$RELBLK1 - Release block, rewrite if dirty
0000      ;
0000      ;  this routine releases the block whose bdb address is in r4.
0000      ;  if the block is dirty it is first written.
0000      ;  if the write is to a disk file, the file must have been extended if needed.
0000      ;
0000      ; Calling sequence:
0000      ;
0000      ;       bsbw    rm$relblk1
0000      ;
0000      ; Input Parameters:
0000      ;
0000      ;       r11     impure area address
0000      ;       r10     ifab address
0000      ;       r9      irab address
0000      ;       r8      rab address
0000      ;       r4      bdb address
0000      ;
0000      ; Implicit Inputs:
0000      ;
0000      ;       bdb$b_flgs                (drt and val)
0000      ;       bdb$b_rel_vbn     -       current block if mbc > 0
0000      ;       bdb$w_numb        -       # bytes to write if magtape
0000      ;       bdb$l_vbn         -       start block for write
0000      ;       bdb$l_addr        -       start address for write
0000      ;       ifb$l_hbk         -       highest allocated block
0000      ;       ifb$w_rtdeq       -       default extend quanity
0000      ;       ifb$w_chnl        -       channel
0000      ;       ifb$l_devbufsiz   -       block buffer size
0000      ;
0000      ; Output Parameters:
0000      ;
0000      ;       r0      status  code
0000      ;       r1,ap   destroyed
0000      ;
0000      ; Implicit Outputs:
0000      ;
0000      ;       bdb$b_flgs        -       drt is cleared
0000      ;                         -       val is set
0000      ;                         -       iop set if write behind underway
0000      ;       bdb$w_numb        -       set to size of transfer, if any
0000      ;
0000      ; Completion Codes:
0000      ;
0000      ;       standard rms, in particular, suc, sys, dme, wer, ful, prv, ext.
0000      ;
0000      ; Side Effects:
0000      ;
0000      ;       may have switched to running at ast level
0000      ;       requiring reprobe of all user addresses not in rab.
0000      ;--
0000
```

```
                        0000     RM$RELBLK1::
                        0000             $TSTPT  RELBLK1
              0C    BB  0006             PUSHR   #^M<R2,R3>              ; save regs.
              01    E1  0008             BBC     #BDB$V_DRT,-
           0A A4        000A                     BDB$B_FLGS(R4),-
              25        000C                     RELEASE                 ; branch if buffer not dirty
              1C    E1  000D             BBC     #DEV$V_RND,-
              6A        000F                     IFB$L_PRIM_DEV(R10),-
              21        0010                     RELEASE                 ; branch if not disk
                        0011
                        0011     ;
                        0011     ; bdb marked dirty     -       adjust bdb data for release
                        0011     ;
                        0011
                        0011             ASSUME  BDB$V_VAL EQ 0
        07 0A A4    E8  0011             BLBS    BDB$B_FLGS(R4),10$      ; branch if buffer valid
                        0015
                        0015     ;
                        0015     ; buffer dirty but marked invalid.
                        0015     ; decrement the current vbn data and say valid.
                        0015     ;
                        0015
           48 A4    97  0015             DECB    BDB$B_REL_VBN(R4)
      0A A4   01    88  0018             BISB2   #BDB$M_VAL,BDB$B_FLGS(R4)
                        001C
                        001C     ;
                        001C     ; adjust byte count for transfer if disk
                        001C     ; (i.e., transfer all blocks thru current vbn or # val_vbns if greater)
                        001C     ;
                        001C
      52   48 A4    9A  001C     10$:    MOVZBL  BDB$B_REL_VBN(R4),R2    , get current vbn
              52    B6  0020             INCW    R2                      ; get # vbns
      49 A4   52    91  0022             CMPB    R2,BDB$B_VAL_VBNS(R4)   ; is current greater than  # valid vbns?
              04    1E  0026             BGEQU   15$                     ; branch if yes
      52   49 A4    9A  0028             MOVZBL  BDB$B_VAL_VBNS(R4),R2   ; no - so use # valid vbns
              52    A5  002C     15$:    MULW3   R2,-
                        002E                     IFB$L_DEVBUFSIZ(R10),-
   14 A4   48 AA        002E                     BDB$W_NUMB(R4)          ; and set xfer size
                        0032
                        0032     ;
                        0032     ; call rm$release to write and release buffer
                        0032     ;
                        0032
                        0032     RELEASE:
         53   02    D0  0032             MOVL    #RLS$M_WRT_THRU,R3      ; flag to get write
            FFC8'   30  0035             BSBW    RM$RELEASE              ; release buffer
              0C    BA  0038             POPR    #^M<R2,R3>
              05        003A             RSB
```

```
                     003B  ;++
                     003B  ; RM$AUTOEXTEND - Extend a sequential disk file
                     003B  ;
                     003B  ;  this subroutine extends a sequential disk file.
                     003B  ;  the # of blocks for the extend is =
                     003B  ;  max (required # blocks,  min (2*bdb$w_numb/512, 256)).
                     003B  ;  if this many blocks are not available only the minimum required are allocated.
                     003B  ;
                     003B  ;  calling sequence:
                     003B  ;
                     003B  ;       bsbw    rm$autoextend
                     003B  ;
                     003B  ;  input parameters:
                     003B  ;
                     003B  ;       r11             impure area address
                     003B  ;       r10             ifab address
                     003B  ;       r9              irab address
                     003B  ;       r8              rab address
                     003B  ;       r4              udb address
                     003B  ;       r2              minimum # of blocks to extend file
                     003B  ;       bdb$w_numb      number of bytes for xfer
                     003B  ;
                     003B  ;  output parameters:
                     003B  ;
                     003B  ;       r0              status code
                     003B  ;       r1,r3,ap        destroyed
                     003B  ;--
                     003B
                     003B  RM$AUTOEXTEND::
            34   BB  003B          PUSHR   #^M<R2,R4,R5>           ; save regs
                     003D
                     003D  ;
                     003D  ; build fib for extend
                     003D  ;
                     003D
    52   40 8F   9A  003D          MOVZBL  #FIB$C_LENGTH,R2        ; size of fib
       51   5A   D0  0041          MOVL    R10,R1                  ; space header addr
         FFB9'   30  0044          BSBW    RM$GETSPC               ; allocate fib
         03 50   E8  0047          BLBS    R0,10$                  ; continue on success
          0081   31  004A          BRW     ERRXIT
    18 A1   6E   D0  004D  10$:    MOVL    (SP),FIB$L_EXSZ(R1)     ; set # blocks required
         1A A1   B5  0051          TSTW    FIB$L_EXSZ+2(R1)        ; hi-order extend size zero?
            08   12  0054          BNEQ    30$                     ; branch if not
    4C AA   6E   B1  0056          CMPW    (SP),IFB$W_RTDEQ(R10)   ; extend size < default?
            05   1E  005A          BGEQU   30$                     ; branch if not
         4C AA   B0  005C          MOVW    IFB$W_RTDEQ(R10),-
         18 A1       005F                  FIB$L_EXSZ(R1)          ; yes - use default
         80 8F   90  0061  30$:    MOVB    #FIB$M_EXTEND,-
         16 A1    .  0064                  FIB$W_EXCTL(R1)         ; say it's an extend
         4C AA   B5  0066          TSTW    IFB$W_RTDEQ(R10)        ; zero default extend size?
            38   12  0069          BNEQ    DOXTND                  ; branch if not
                     006B
                     006B  ;
                     006B  ;  This is an auto extend with deq=0.
                     006B  ;  Try to use system or process default extention quantity.
                     006B  ;  Otherwise, compute the number of blocks to extend =
                     006B  ;       max(required,min(2*bufsiz,256))
```

```
                                  006B      ;
                                  006B
          50   00000000'9F   3C   006B               MOVZWL  @#PIO$GW_RMSEXTEND,R0      ; Process deq?
                         22   12  0072               BNEQ    33$                         ; branch if yes
          50   00000000'9F   3C   0074               MOVZWL  @#SYS$GW_RMSEXTEND,R0      ; System deq?
                         19   12  007B               BNEQ    33$                         ; branch if yes
               54    04  AE   D0  007D               MOVL    4(SP),R4                    ; restore bdb address
  50    14 A4    07      09   EF  0081               EXTZV   #9,#7,BDB$W_NUMB(R4),R0    ; get # of blocks to xfer
                 50      02   A4  0087               MULW2   #2,R0                       ; times 2
          0100 8F        50   B1  008A               CMPW    R0,#256                     ; > max. default extend size?
                         05   15  008F               BLEQ    33$                         ; branch if not
          50   0100 8F        B0  0091               MOVW    #256,R0                     ; yes - just extend 256 blocks
               6E        50   D1  0096      33$:     CMPL    R0,(SP)                     ; > required extend size?
                         08   1B  0099               BLEQU   36$                         ; branch if not
            18 A1        50   B0  009B               MOVW    R0,FIB$L_EXSZ(R1)           ; use larger default extend size
                                  009F               SSB     #IFB$V_TEF,(R10)            ; and say used so that we truncate
                                  00A3                                                   ; at eof on close
                                  00A3      36$:
                                  00A3
                                  00A3      ;
                                  00A3      ; build the fib descriptor
                                  00A3      ;
                                  00A3
                         51   DD  00A3      DOXTND: PUSHL   R1                          ; addr of fib
            7E        40 8F   9A  00A5               MOVZBL  #FIB$C_LENGTH,-(SP)         ; length of fib
                                  00A9
                                  00A9      ;
                                  00A9      ; do the extend
                                  00A9      ;
                                  00A9
                      FF54'   30  00A9               BSBW    RM$FCPEXTEND                ; & do extend
                 53      5A   D0  00AC               MOVL    R10,R3                      ; get set for rm$retspc
                         14   BA  00AF               POPR    #^M<R2,R4>                  ; restore fib length (r2) and addr (r4)
          03EC 8F        50   B1  00B1               CMPW    R0,#SS$_EXDISKQUOTA         ; hit overdraft ?
                         37   13  00B6               BEQLU   OVERQUOTA                   ; spec handling if overquota
                                  00B8
          0850 8F        50   B1  00B8               CMPW    R0,#SS$_DEVICEFULL          ; if device full -
                         15   13  00BD               BEQLU   RETRY                       ;    retry
                                  00BF
                      21 50   E9  00BF               BLBC    R0,ERREXT                   ; if error code, exit
```

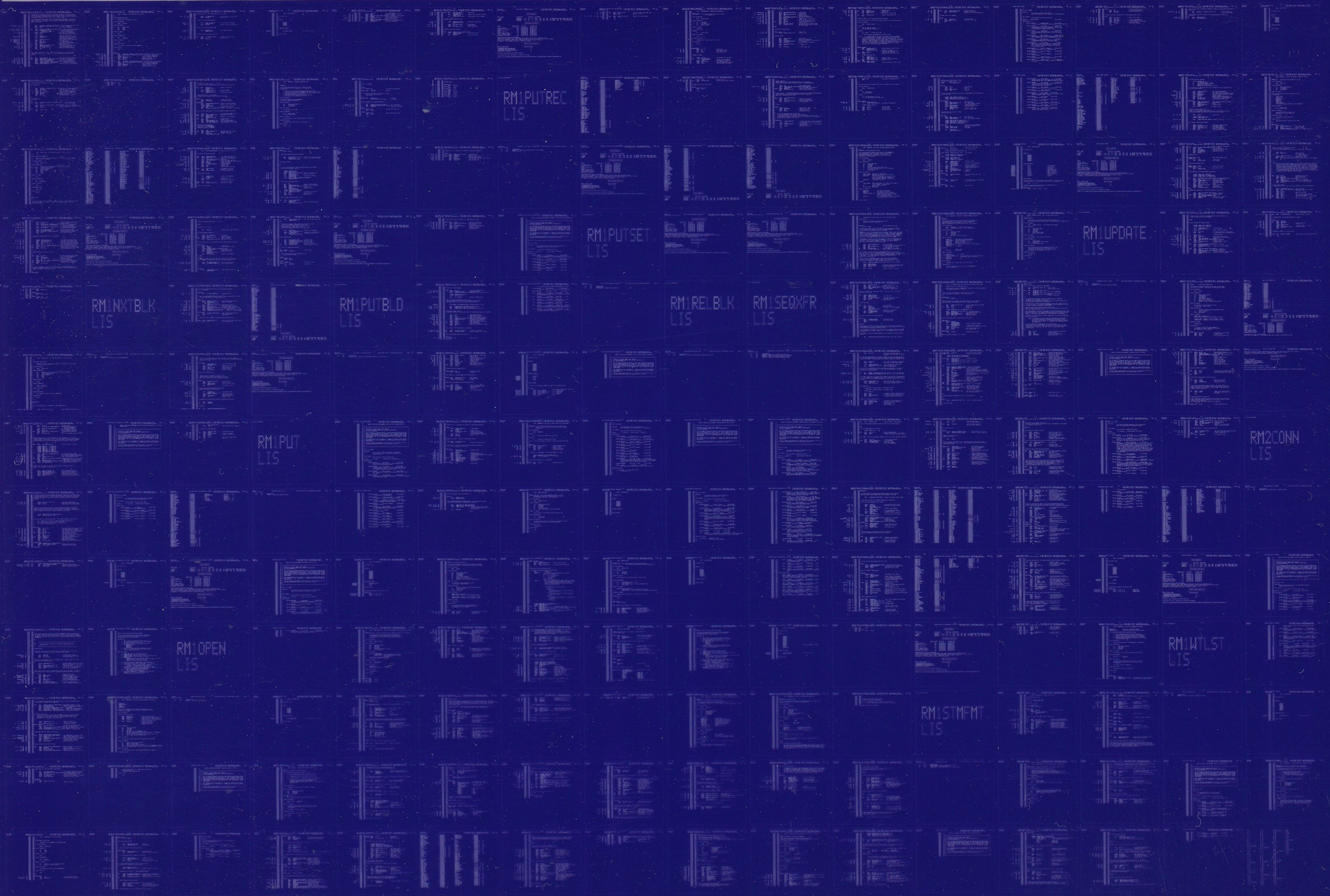
```
                              00C2
                              00C2      ;
                              00C2      ; extend complete.
                              00C2      ; update ifab hi block field, deallocate the fib, and return
                              00C2      ; return status can either be straight success, or SS$_OVRDSKQUOTA
                              00C2      ;
                              00C2
                     50  DD   00C2              PUSHL   R0                              ; save success status
  70 AA   1C A4   18 A4  C1   00C4              ADDL3   FIB$L_EXSZ(R4),FIB$L_EXVBN(R4),IFB$L_HBK(R10)
                  70 AA  D7   00CB              DECL    IFB$L_HBK(R10)                  ; get # of highest allocated blk
                  FF2F'  30   00CE      ERRXIT: BSBW    RM$RETSPC                       ; return the fib space
                     35  BA   00D1              POPR    #^M<R0,R2,R4,R5>                ; restore status and regs
                         05   00D3              RSB
                              00D4
                              00D4      ;
                              00D4      ; extend failed.  if attempted extend was for more than minimum required amount,
                              00D4      ; retry extend for only the minimum required amount, otherwise
                              00D4      ; map error, release buffer & return
                              00D4      ;
                              00D4
              18 A4      6E  D1   00D4  RETRY:  CMPL    (SP),FIB$L_EXSZ(R4)             ;
                         09  1E   00D8          BGEQU   ERREXT                          ; branch if leq minimum
                   51    54  D0   00DA          MOVL    R4,R1                           ; restore fib address
              18 A1      6E  D0   00DD          MOVL    (SP),FIB$L_EXSZ(R1)             ; must extend this amount
                         C0  11   00E1          BRB     DOXTND                          ; go retry extend
                              00E3
                              00E3      ;
                              00E3      ;  errors
                              00E3      ;
                              00E3
                              00E3      ERREXT:
                              00E3              RMSERR  EXT,R1                          ; default error code
                  FF15'  30   00E8              BSBW    RM$MAPERR                       ; map the error code
                     50  DD   00EB              PUSHL   R0                              ; save error code
                     DF  11   00ED              BRB     ERRXIT                          ; cleanup and return
                              00EF
                              00EF      ;
                              00EF      ;   we hit the over-quota mark on the disk.
                              00EF      ;   if the user asked for this amount, then tell him.
                              00EF      ;   if we guessed a value to extend by due to auto-extend, then tell him
                              00EF      ;   but set the the amount to use from now on as 1 so as not to tie up blocks
                              00EF      ;   now that we are close to the limit.
                              00EF      ;
                              00EF
                              00EF      OVERQUOTA:
                  4C AA  B5   00EF              TSTW    IFB$W_RTDEQ(R10)                ; if zero, then this is first time we're
                              00F2                                                      ; in overdraft and we did auto-extend
                     EF  12   00F2              BNEQ    ERREXT                          ; no - go away
                  4C AA  B6   00F4              INCW    IFB$W_RTDEQ(R10)                ; default extend is now 1, this disables
                              00F7                                                      ; auto-extend
                     EA  11   00F7              BRB     ERREXT                          ; return to mainline
                              00F9
                              00F9              .END
```

```
$$.PSECT_EP                      = 00000000
$$RMSTEST                        = 0000001A
$$RMS_PBUGCHK                    = 00000010
$$RMS_TBUGCHK                    = 00000008
$$RMS_UMODE                      = 00000004
BDB$B_FLGS                       = 0000000A
BDB$B_REL_VBN                    = 00000048
BDB$B_VAL_VBNS                   = 00000049
BDB$M_VAL                        = 00000001
BDB$V_DRT                        = 00000001
BDB$V_VAL                        = 00000000
BDB$W_NUMB                       = 00000014
DEV$V_RND                        = 0000001C
DOXTND                             000000A3 R      01
ERREXT                             000000E3 R      01
ERRXIT                             000000CE R      01
FIB$C_LENGTH                     = 00000040
FIB$L_EXSZ                       = 00000018
FIB$L_EXVBN                      = 000^001C
FIB$M_EXTEND                     = 00000080
FIB$W_EXCTL                      = 00000016
IFB$L_DEVBUFSIZ                  = 00000048
IFB$L_HBK                        = 00000070
IFB$L_PRIM_DEV                   = 00000000
IFB$V_TEF                        = 00000036
IFB$W_RTDEQ                      = 0000004C
OVERQUOTA                          000000EF R      01
PIO$A_TRACE                        ********    X   01
PIO$GQ_RMSEXTEND                   ********    X   01
RELEASE                            00000032 R      01
RETRY                              000000D4 R      01
RLS$M_WRT_THRU                   = 00000002
RM$AUTOEXTEND                      0000003B RG     01
RM$FCPEXTEND                       ********    X   01
RM$GETSPC                          ********    X   01
RM$MAPERR                          ********    X   01
RM$RELBLK1                         00000000 RG     01
RM$RELEASE                         ********    X   01
RM$RETSPC                          ********    X   01
RMS$_EXT                         = 0001C022
SS$_DEVICEFULL                   = 00000850
SS$_EXDISKQUOTA                  = 000003EC
SYS$GW_RMSEXTEND                   ********    X   01
TPT$L_RELBLK1                      ********    X   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
RM$RMS1                          000000F9  (  249.)  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                   30     00:00:00.10    00:00:01.07
Command processing              109     00:00:00.73    00:00:04.50
Pass 1                          341     00:00:11.61    00:00:33.22
Symbol table sort                 0     00:00:01.84    00:00:02.17
Pass 2                           80     00:00:02.18    00:00:05.37
Symbol table output               7     00:00:00.09    00:00:00.12
Psect synopsis output             2     00:00:00.03    00:00:00.02
Cross-reference output            0     00:00:00.00    00:00:00.00
Assembler run totals            571     00:00:16.58    00:00:46.57

The working set limit was 1500 pages.
66017 bytes (129 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1321 non-local and 7 local symbols.
414 source lines were read in Pass 1, producing 13 object records in Pass 2.
22 pages of virtual memory were used to define 21 macros.

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

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                       10
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                        0
_$255$DUA28:[SYSLIB]STARLET.MLB;2                     7
TOTALS (all libraries)                               17

1439 GETS were required to define 17 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:RM1RELBLK/OBJ=OBJ$:RM1RELBLK MSRC$:RM1RELBLK/UPDATE=(ENH$:RM1RELBLK)+EXECML$/LIB+LIB$:RMS/LIB
```

0322 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



RM1PUTREC
LIS

RM1PUTSET
LIS

RM1UPDATE
LIS

RM1NXTBLK
LIS

RM1PUTBLD
LIS

RM1RELBLK
LIS

RM1SEQXFR
LIS

RM1PUT
LIS

RM2CONN
LIS

RM1OPEN
LIS

RM1WTLST
LIS

RM1STMFMT
LIS