```
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
```

FILEID**RMSORNDWN

RMS
V04

```
RRRRRRRR    MM        MM    SSSSSSSS     000000     RRRRRRRR    NN        NN  DDDDDDDD      WW        WW  NN        NN
RRRRRRRR    MM        MM    SSSSSSSS     000000     RRRRRRRR    NN        NN  DDDDDDDD      WW        WW  NN        NN
RR      RR  MMMM    MMMM  SS           00      00   RR      RR  NN        NN  DD      DD    WW        WW  NN        NN
RR      RR  MMMM    MMMM  SS           00      00   RR      RR  NN        NN  DD      DD    WW        WW  NN        NN
RR      RR  MM  MM  MM    SS           00    0000   RR      RR  NNNN      NN  DD      DD    WW        WW  NNNN      NN
RR      RR  MM  MM  MM    SS           00    0000   RR      RR  NNNN      NN  DD      DD    WW        WW  NNNN      NN
RRRRRRRR    MM        MM    SSSSSS     00  00  00   RRRRRRRR    NN  NN    NN  DD      DD    WW        WW  NN  NN    NN
RRRRRRRR    MM        MM    SSSSSS     00  00  00   RRRRRRRR    NN  NN    NN  DD      DD    WW        WW  NN  NN    NN
RR  RR      MM        MM          SS   0000    00   RR  RR      NN    NNNN    DD      DD    WW  WW    WW  NN    NNNN
RR  RR      MM        MM          SS   0000    00   RR  RR      NN    NNNN    DD      DD    WW  WW    WW  NN    NNNN
RR    RR    MM        MM          SS   00      00   RR    RR    NN        NN  DD      DD    WWWW  WWWW    NN        NN
RR    RR    MM        MM          SS   00      00   RR    RR    NN        NN  DD      DD    WWWW  WWWW    NN        NN      ....
RR      RR  MM        MM  SSSSSSSS       000000     RR      RR  NN        NN  DDDDDDDD      WW        WW  NN        NN      ....
RR      RR  MM        MM  SSSSSSSS       000000     RR      RR  NN        NN  DDDDDDDD      WW        WW  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
```

```
0000          $BEGIN  RMSORNDWN,001,RM$RMS,<RMS IO RUN DOWN>
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: this module insures all rms i/o activity is complete,
0000    ;           closes all files, and resets the ifab and irab tables.
0000    ;
0000    ; Environment:
0000    ;                   star processor running starlet exec.
0000    ;
0000    ; Author: l f laverdure,          creation date: 5-5-77
0000    ;
0000    ; Modified By:
0000    ;
0000    ;       V04-001 RAS0332         Ron Schaefer            14-Sep-1984
0000    ;               ALWAYS re-enable ASTS when stalling inside rundown
0000    ;               as they could get disabled by the previous exec mode
0000    ;               thread of RMS that will never continue and re-enable them.
0000    ;
0000    ;       V03-005 DGB0040         Donald G. Blair         02-May-1984
0000    ;               If the PIO$V_INHAST bit is set when we start an
0000    ;               RMS operation, we conclude that the caller must be
0000    ;               at exec AST level or higher and would break RMS
0000    ;               synchronization rules if he were allowed to continue.
0000    ;               Return error.  This fix also includes a change from
0000    ;               Jim Johnson to clear the FID correctly in GETDVIFID.
0000    ;
0000    ;       V03-004 SHZ0001         Stephen H. Zalewski     14-Sep-1983
0000    ;               Move routine RM$GETDVIFID from module RMOGETDVI to here, and
0000    ;               rename it GETDVIFID.  Module RMOGETDVI has been evaporated.
0000    ;
0000    ;       V03-003 JWH0107         Jeffrey W. Horn         24-Sep-1982
```

```
0000    ;               Add call to RMS$RU_UNLOCK to release locks
0000    ;               held for the duration of a recovery unit.
0000    ;
0000    ;       V03-002 KBT0316         Keith B. Thompson       8-Sep-1982
0000    ;               Remove all S0 sharing code
0000    ;
0000    ;       V03-001 KBT0191         Keith B. Thompson       23-Aug-1982
0000    ;               Reorganize psects and rename entry points to single '$'
0000    ;
0000    ;--
0000
```

```
0000          .SBTTL  DECLARATIONS
0000
0000      ;
0000      ; Include Files:
0000      ;
0000
0000      ;
0000      ; Macros:
0000      ;
0000
0000          $DEVDEF
0000          $FABDEF
0000          $FIBDEF
0000          $FWADEF
0000          $IFBDEF
0000          $IRBDEF
0000          $IMPDEF
0000          $NWADEF
0000          $PIODEF
0000          $PSLDEF
0000          $RLBDEF
0000          $RMSDEF
0000
0000      ;
0000      ; Equated Symbols:
0000      ;
0000
0000      ;
0000      ; Own Storage:
0000      ;
0000
```

```
0000          .SBTTL  RMS$RMSRUNDWN - RMS I/O RUN DOWN
0000
0000    ;++
0000    ; RMS$RMSRUNDWN - RMS I/O run down
0000    ;
0000    ; this routine first determines the type of rundown desired, based
0000    ; upon the second argument.  if the type is "abort rms i/o", a branch
0000    ; is made to rm$last_chance, otherwise the routine checks that all ifabs and irabs
0000    ; are inactive.  if any found active this routine awaits their completion after
0000    ; first performing a $cancel i/o if not a file-oriented device.
0000    ; when all i/o activity for the file is complete, $close is
0000    ; performed for the file.  if the close failed for an output file
0000    ; on a files-oreiented device, an error is returned to the caller
0000    ; who should note the error and recall this routine to run down
0000    ; further files.  if all files are successfully run down the
0000    ; image ifab & irab tables are reset and return is made to the
0000    ; caller with a success code.
0000    ;
0000    ; files are run down in this order:
0000    ;
0000    ;         1.  indirect process permanent files
0000    ;             ('error' should be first)
0000    ;         2.  image files
0000    ;         3.  (only if caller's mode is not user and arg2=1)
0000    ;             process permanent files
0000    ;
0000    ; Calling sequence:
0000    ;
0000    ;         calls   #2, sys$rmsrundwn
0000    ;
0000    ; Input Parameters:
0000    ;
0000    ;         ap      users argument list (2 arguments)
0000    ;
0000    ;         arg1    descriptor for 22-character buffer
0000    ;                 to receive information about
0000    ;                 unsuccessfully closed output file
0000    ;                 (device id and file id)
0000    ;         arg2    rundown type, as follows:
0000    ;
0000    ;                 0 - run down of image and indirect i/o for process permanent files
0000    ;                 1 - run down of image and process permanent files
0000    ;                     (caller's mode must be other than user)
0000    ;                 2 - abort rms i/o (caller's mode must be exec or kernel)
0000    ;
0000    ;                 all others are reserved, but currently behave as type 0
0000    ;
0000    ; Implicit Inputs:
0000    ;
0000    ;         caller's mode.
0000    ;
0000    ; Output Parameters:
0000    ;
0000    ;         r0      status code
0000    ;         r1      destroyed
0000    ;
0000    ; Implicit Outputs:
```

```
0000  ;
0000  ; information describing an output file unsuccessfully closed is
0000  ; stored in the caller-provided buffer in exactly the same
0000  ; format as the dvi, did, and fid fields of the nam block.
0000  ;
0000  ; Completion Codes:
0000  ;
0000  ;       standard rms, in particular:
0000  ;
0000  ;       rms$_suc  -  all files closed
0000  ;       rms$_ccf  -  an output file could not be closed
0000  ;                    successfully.  caller-provided buffer
0000  ;                    has information identifying the file
0000  ;       rms$_ial  -  same as rms$_ccf except could not
0000  ;                    access caller's buffer to store file
0000  ;                    id information.
0000  ;
0000  ; Side Effects:
0000  ;
0000  ;       runs synchronously in exec mode inhibiting
0000  ;       and enabling asts as required.
0000  ;
0000  ;--
0000
```

```
                            0000          $ENTRY  RMS$RMSRUNDWN
                            0000          $TSTPT  RUNDWN
                 5B   DC    0006          MOVPSL  R11
                 16   EF    0008          EXTZV   #PSL$V_PRVMOD,-
        57   5B  02         000A                  #PSL$S_PRVMOD,R11,R7      ; save caller's mode
         02   08 AC   D1    000D          CMPL    8(AP),#2                  ; abort rms i/o?
                 42   13    0011          BEQL    RMSABORT                  ; branch if yes
                 00   E2    0013          BBSS    #PIO$V_INHAST,-           ; br if RMS already in progress
      40 00000000'9F        0015                  @#PIO$GW_STATUS,ERRBUSY
                            001B
                            001B    ;
                            001B    ; start by releasing locks held for the durration of a recovery unit,
                            001B    ; if any.
                            001B    ;
                            001B
              FFE2'   30    001B          BSBW    RM$RU_UNLOCK
                            001E
                            001E    ;
                            001E    ; next run down indirect i/o on  process-permanent files
                            001E    ;
                            001E
   5B    00000000'9F  DE    001E          MOVAL   @#PIO$GW_PIOIMPA,R11      ; get pio impure area address
                            0025          ASSUME  IMP$W_RMSSTATUS EQ 0
            58   01   D0    0025          MOVL    #1,R8                     ; indicate indirect run down
              006B    30    0028          BSBW    RUNDWN                    ; do the run down
                            002B                                            ; (note: clears r8)
                            002B    ;
                            002B    ; now run down the image
                            002B    ;
                            002B
      5B    0000'CB   DE    002B          MOVAL   W^PIO$GW_IIOIMPA-PIO$GW_PIOIMPA(R11),R11
                            0030
                            0030    ;
                            0030    ; point to image impure area
                            0030    ;
                            0030
              0063    30    0030          BSBW    RUNDWN                    ; do the run down
             18 BB    D4    0033          CLRL    @IMP$L_IFABTBL(R11)       ; reset ifab table link
             1C BB    D4    0036          CLRL    @IMP$L_IRABTBL(R11)       ; reset irab table link
                            0039
                            0039    ;
                            0039    ; point to process
                            0039    ; i/o impure area again
                            0039    ;
                            0039
      5B    0000'CB   DE    0039          MOVAL   W^PIO$GW_PIOIMPA-PIO$GW_IIOIMPA(R11),R11
                            003E    ;
                            003E    ; At this point there used to be code to return any whole pages
                            003E    ; on the FMLH free space list back to the process i/o free page
                            003E    ; list.  The space on the FMLH list is currently (v 2) used only
                            003E    ; for ASB allocation on IFAB operations and will bugcheck if space
                            003E    ; is not found.  The behavior is now that a page will be added to
                            003E    ; the FMLH list the first time a process stalls on an IFAB operation
                            003E    ; and will remain there for the life of the process.
                            003E    ;
                            003E
                            003E    ;
```

```
                          003E   241 ; now run down direct i/o on process-permanent files if desired
                          003E   242 ;
                          003E   243
   01    08 AC      91    003E   244         CMPB    8(AP),#1                ; ppf rundown?
            07      12    0042   245         BNEQ    XITSUC                  ; branch if not
      03    57      91    0044   246         CMPB    R7,#PSL$C_USER          ; caller sufficiently privileged?
            02      13    0047   247         BEQL    XITSUC                  ; branch if not
            4B      10    0049   248 60$:    BSBB    RUNDWN                  ; do the run down
                          004B   249 XITSUC: RMSSUC
                          004E   250 EXIT:
                          004E   251         SSB     #16, R0                 ; stamp 'rms' on status code
            11      10    0052   252         BSBB    ENBAST                  ; enable asts
                    04    0054   253         RET                             ; back to caller
                          0055   254
                          0055   255 ;
                          0055   256 ;  branch to rm$last_chance to do async process deletion rms i/o abort
                          0055   257 ;
                          0055   258
                          0055   259 RMSABORT:
  00000000'EF       17    0055   260         JMP     RM$LAST_CHANCE
```

```
                          005B  ;
                          005B  ;  If the PIO$V_INHAST bit is already set, we
                          005B  ;  conclude that the caller must be at exec ast level or higher
                          005B  ;  (otherwise, he could not have kicked off an RMS operation
                          005B  ;  while RMS was already in progress) and would break RMS
                          005B  ;  synchronization rules if allowed to continue.  Return RMS$_BUSY
                          005B  ;  status when this happens.
                          005B  ;
                          005B
                          005B  ERRBUSY:
                          005B          RMSERR  BUSY
                          0060          SSB     #16,R0
                      04  0064          RET
                          0065
                          0065  ;
                          0065  ; enable rms ast's, reenabling exec ast's in all cases.
                          0065  ;
                          0065
                          0065  ENBAST: CSB     #PIO$V_INHAST, @#PIO$GW_STATUS
                          006D
                          006D  ;
                          006D  ; clear ast inhibit and enable asts
                          006D  ;
                          006D
                          006D          $SETAST_S       #1              ; enable exec mode asts
                      05  0076          RSB
                          0077
                          0077  ;
                          0077  ; inhibit rms asts
                          0077  ;
                          0077
00000000'9F     01    A8  0077  INHAST: BISW2   #1@PIO$V_INHAST, @#PIO$GW_STATUS
                      05  007E          RSB
                          007F
                          007F  ;
                          007F  ; wait for rms operation completion
                          007F  ;
                          007F
                          007F  WAIT:   $CLREF_S        #IMP$C_IOREFN   ; clear rms event flag
                DB    10  0088          BSBB    ENBAST                  ; enable asts
                          008A          $WAITFR_S       #IMP$C_IOREFN   ; wait for flag
                E2    10  0093          BSBB    INHAST                  ; re-inhibit asts
                      05  0095          RSB
```

```
                     0096
                     0096  ;++
                     0096  ;
                     0096  ; run down subroutine:
                     0096  ;
                     0096  ; checks ifab table for active files.
                     0096  ; if any found waits for any i/o activity to finish
                     0096  ; (doing a cancell i/o for non files-oriented devices)
                     0096  ; and then issues a $close request.
                     0096  ;
                     0096  ; when all files run down performs a sanity check by seeing if all irab
                     0096  ; table entries are also zero.
                     0096  ;
                     0096  ; inputs:
                     0096  ;       r11 - impure area addr
                     0096  ;       r8 - bit 0 set if indirect ppf run down
                     0096  ;       ap - caller's arg list
                     0096  ;       r7 - caller's mode
                     0096  ;
                     0096  ; outputs:
                     0096  ;       returns only if noerror encountered.
                     0096  ;       imp$v_ppfindrd cleared
                     0096  ;       r0 - r6, r9, r10 destroyed
                     0096  ;--
                     0096
                     0096  RUNDWN:
                     0096          SSB     #IMP$V_IORUNDOWN,(R11)  ; set i/o rundown in progress flag
                     009A                                          ; to sync with ast-driven rms
                     009A                                          ; operations
 55   18 AB    D0    009A          MOVL    IMP$L_IFABTBL(R11),R5   ; get ifab table addr
         56    D4    009E          CLRL    R6                      ; build ifi value here
    52   85    D0    00A0  NXTSEG: MOVL    (R5)+,R2                ; save addr next table seg in r2
 54   20 AB    3C    00A3          MOVZWL  IMP$W_ENTPERSEG(R11),R4 ; get # entries/seg
      22 AB    B5    00A7  NXTENT: TSTW    IMP$W_NUM_IFABS(R11)    ; any ifabs active?
         0F    13    00AA          BEQL    CHKIRB                  ; branch if none
         56    D6    00AC          INCL    R6                      ; bump ifi
    5A   85    D0    00AE          MOVL    (R5)+,R10               ; get ifab addr
         2F    12    00B1          BNEQ    RDIFAB                  ; branch if one
      F1 54    F5    00B3  NXTSOB: SOBGTR  R4,NXTENT               ; keep scanning segment
                     00B6
                     00B6  ;
                     00B6  ; no more ifabs this segment, try next
                     00B6  ;
                     00B6
 55      52    D0    00B6          MOVL    R2,R5                   ; get next segment addr
         E5    12    00B9          BNEQ    NXTSEG                  ; branch if one
```

```
                     00BB
                     00BB      ;
                     00BB      ; all ifabs have been run down now.
                     00BB      ;
                     00BB      ; unless this is indirect run down of ppf's,
                     00BB      ; check that all irabs are also gone.
                     00BB      ;
                     00BB
                     00BB      CHKIRB:
          00000008   00BB              .IF     NE $$RMSTEST&$$RMS_TBUGCHK
17 58    00    E4    00BB              BBSC    #0,R8,30$                  ; branch if indirect run down
55    1C AB    D0    00BF              MOVL    IMP$L_IRABTBL(R11),R5      ; get irab table addr
   52    85    D0    00C3      10$:    MOVL    (R5)+,R2                   ; save addr next table seg.
54    20 AB    3C    00C6              MOVZWL  IMP$W_ENTPERSEG(R11),R4    ; get # entries/seg.
         85    D5    00CA      20$:    TSTL    (R5)+                      ; entry zero?
         0D    12    00CC              BNEQ    ERRBUG                     ; branch if not
      F9 54    F5    00CE              SOBGTR  R4,20$                     ; branch if more entries
   55    52    D0    00D1              MOVL    R2,R5                      ; get next seg addr
         ED    12    00D4              BNEQ    10$                        ; branch if one
                     00D6      30$:    CSB     #IMP$V_IORUNDOWN,(R11)     ; turn off rundown in progress flag
                     00DA              .ENDC
               05    00DA              RSB                                ; all o.k.
                     00DB
                     00DB      ;
                     00DB      ; close failed to zero ifab or irab table entry
                     00DB      ;
                     00DB      ERRBUG: RMSTBUG FTL$_IORNDN
```

```
                          00E2   381
                          00E2   382 ;
                          00E2   383 ; found an ifab.  check for active and if so allow operation to finish
                          00E2   384 ;
                          00E2   385
                          00E2   386          ASSUME  IMP$W_RMSSTATUS EQ 0
                          00E2   387 RDIFAB:
       2A 58      E8      00E2   388          BLBS    R8,RDNET                ; branch if indirect ppf
    26 6A   20    E1      00E5   389 10$:     BBC     #IFB$V_BUSY,(R10),RDNET ; if not busy then check NETWORK
    04 6A   0D    E0      00E9   390          BBS     #DEV$V_NET,(R10),20$    ; do cancel if busy & network operation
            3A    E1      00ED   391          BBC     #IFB$V_RMS_STALL,-      ; if this RMS thread is not currently
       30 6A              00EF   392                  (R10),RDIRAB            ; stalled then skip the cancel and wait
                          00F1   393
                          00F1   394 ;
                          00F1   395 ; allow function to finish
                          00F1   396 ; \note: this code should be modified to
                          00F1   397 ; properly run down read-ahead and write-behind
                          00F1   398 ; operations to unit record devices.\
                          00F1   399 ;
                          00F1   400
                          00F1   401 20$:     $CANCEL_S       IFB$W_CHNL(R10) ; cancel i/o (e.g. magtape create)
            81    10      00FC   402          BSBB    WAIT                    ; wait for an operation to finish
       FC A5      D5      00FE   403          TSTL    -4(R5)                  ; ifab disappear? (close)
            B0    13      0101   404          BEQL    NXTSOB                  ; branch if yes
    08 6A   20    E1      0103   405          BBC     #IFB$V_BUSY,(R10),RDNET ; run down NETWORK if no longer busy
    E6 6A   0D    E0      0107   406          BBS     #DEV$V_NET,(R10),20$    ; but do cancel & wait again if busy &
            3A    E0      010B   407          BBS     #IFB$V_RMS_STALL,-      ; network operation or busy and the RMS
       E2 6A              010D   408                  (R10),20$               ; thread is still stalled
                          010F   409
                          010F   410 ;
                          010F   411 ; if the current operation is a network operation, and a special recieve QIO
                          010F   412 ; has been posted but NOT recieved, a $CANCEL must always be done to flush
                          010F   413 ; this QIO. In file transfer mode it will be possible that a recieve has been
                          010F   414 ; posted but no transfer operation is underway. therefore neither the IFAB nor
                          010F   415 ; the IRAB will be busy. if a $CANCEL isn't explicitely issued, when the $CLOSE
                          010F   416 ; is performed, the NETDRIVER will be unable to disconnect the logical link
                          010F   417 ; (because of the outstanding recieve), and the process will hang.
                          010F   418 ;
                          010F   419
    0E 6A   0D    E1      010F   420 RDNET:   BBC     #DEV$V_NET,(R10),RDIRAB ; go run down IRABs if not network op
    50   3C AA    D0      0113   421          MOVL    IFB$L_NWA_PTR(R10),R0   ; obtain network work area address
            08    13      0117   422          BEQL    RDIRAB                  ; skip check if not network work area
            03    E1      0119   423          BBC     #NWA$V_RCVQIO,-         ; if a special recieve QIO has not been
       04 60              011B   424                  (R0),RDIRAB             ; posted go run down the IRABs, but if
            04    E1      011D   425          BBC     #NWA$V_RCVAST,-         ; one has and it hasn't been recieved
       29 60              011F   426                  (R0),CANCEL             ; then go issue the cancel
                          0121   427
                          0121   428 ;
                          0121   429 ; run down irabs
                          0121   430 ;
                          0121   431
                          0121   432 RDIRAB:
       59   5A    D0      0121   433          MOVL    R10,R9                  ; copy ifab addr
    59   1C A9    D0      0124   434 10$:     MOVL    IRB$L_IRAB_LNK(R9),R9   ; get next irab
            30    13      0128   435          BEQL    QUIET                   ; branch if none
       03 58      E8      012A   436          BLBS    R8,12$                  ; don't release locks if indirect PPF
                          012D   437                                          ; rundown
```

```
           FEDO'  30  012D   438          BSBW     RMS$UNLOCKALL              ; kill all record locks, including
                      0130   439                                              ; outstanding waits.
   F0 69    20    E1  0130   440 12$:     BBC      #IRB$V_BUSY,(R9),10$       ; branch if idle
   04 6A    0D    E0  0134   441          BBS      #DEV$V_NET,(R10),15$       ; do cancel if busy & network operation
            3A    E1  0138   442          BBC      #IRB$V_RMS_STALL,-         ; if this RMS thread is not currently
      E8 69           013A   443                   (R9),10$                   ; stalled then skip the cancel and wait
                      013C   444
      07 6B       E8  013C   445 15$:     BLBS     (R11), 20$                 ; branch if image i/o segment
   03 69    22    E0  013F   446          BBS      #IRB$V_PPF_IMAGE,(R9),20$
                      0143   447
                      0143   448 ;
                      0143   449 ; branch if indirect i/o
                      0143   450 ;
                      0143   451
      DE 58       E9  0143   452          BLBC     R8,10$                     ; branch if only indirect ppfs
                      0146   453                                              ; to be run down
            1C    E0  0146   454 20$:     BBS      #DEV$V_RND,-
            6A        0148   455                   IFB$L_PRIM_DEV(R10),-      ; no need to do a cancel if this is
            0B        0149   456                   NOCANCEL                   ; a disk operation, just go wait
                      014A   457
                      014A   458 CANCEL: $CANCEL_S       IFB$W_CHNL(R10)      ; cancel i/o
                      0155   459
                      0155   460 NOCANCEL:
           FF27   30  0155   461          BSBW     WAIT                       ; wait for all ASTs to be delivered
            C7    11  0158   462          BRB      RDIRAB                     ; start from top of irab
                      015A   463                                              ; chain again (could
                      015A   464                                              ; have been disconnect)
```

```
                               015A
                               015A    ;
                               015A    ; all activity ceased for this file.
                               015A    ; force a close by constructing a fab and calling close.
                               015A    ;
                               015A
          78 AA        D5      015A    QUIET:  TSTL     IFB$L_SFSB_PTR(R10)             ; is it a shared file?
             0C        12      015D            BNEQ     5$                              ; yes, go close it
       35 6A 30        E1      015F            BBC      #IFB$V_WRTACC,(R10),NOERR       ; branch if not write access
       31 6A 03        E1      0163            BBC      #DEV$V_DIR,IFB$L_PRIM_DEV(R10),NOERR
       2D 6A 25        E1      0167            BBC      #IFB$V_ACCESSED,(R10),NOERR     ; branch if file not accessed
       50   04 AC      D0      016B    5$:     MOVL     4(AP),R0                        ; get descriptor addr
                               016F            IFNORD   #8, (R0), NOERR1, R7
          1C   60      B1      0175            CMPW     (R0), #28                       ; at least 22 bytes long?
               17      1F      0178            BLSSU    NOERR1
       53   04 A0      D0      017A            MOVL     4(R0),R3                  ; get buffer address
          59   5A      D0      017E            MOVL     R10,R9                    ; ifab to right register
                               0181            IFNOWRT  #22,(R3),NOERR1,R7        ; branch if buffer not writable
             0069      30      0187            BSBW     GETDVIFID                 ; go fill buffer with dvi and fid
                               018A            RMSERR   CCF,R3                    ; get set for close failure
               0A      11      018F            BRB      CLOSE
                               0191
                               0191    NOERR1: RMSERR   IAL,R3                    ; if close failure, return ial
               03      11      0196            BRB      CLOSE
                               0198
                               0198    NOERR:  RMSSUC   SUC,R3                    ; can't fail
                               019B
       5E   B0 AE      DE      019B    CLOSE:  MOVAL    -FAB$C_BLN(SP),SP         ; create fab on stack
          5003 8F      B0      019F            MOVW     #FAB$C_BID+<FAB$C_BLN @8>,-
               6E              01A3                     (SP)                      ; fab block id and length
       02 AE   56      B0      01A4            MOVW     R6, FAB$W_IFI(SP)         ; ifi
          0B 6B        E8      01A8            BLBS     (R11),10$                 ; branch if iio seg
                               01AB            SSB      #15+<FAB$W_IFI*8>,(SP)    ; set pio flag
          04 58        E9      01AF            BLBC     R8,10$                    ; branch if direct access
                               01B2            SSB      #FAB$V_PPF_IND+<FAB$W_IFI*8>,-
                               01B2                     (SP)                      ; else make indirect ifi
               3C      BB      01B6    10$:    PUSHR    #^M<R2,R3,R4,R5>          ; save regs
     00     6E 00      2C      01B8            MOVC5    #0,(SP),#0,-
   14 AE   004C 8F             01BC                     #FAB$C_BLN-4,4+<4*4>(SP); zero remainder of fab
               3C      BA      01C1            POPR     #^M<R2,R3,R4,R5>          ; restore r5
             FE9F      30      01C3            BSBW     ENBAST
               5E      DD      01C6            PUSHL    SP                        ; addr of fab
  00000000'9F  01      FB      01C8            CALLS    #1,@#SYS$CLOSE            ; close it
             FEA5      30      01CF            BSBW     INHAST
  5E   00000050 8F     C0      01D2            ADDL     #FAB$C_BLN,SP             ; 'pop' fab
            05 58      E8      01D9            BLBS     R8,15$                    ; omit check if indirect ppf
            FC A5      D5      01DC            TSTL     -4(R5)                    ; did ifab go away?
               0F      12      01DF            BNEQ     ERRBUG_BR                 ; branch if not
            03 50      E9      01E1    15$:    BLBC     R0,30$                    ; branch on error
             FECC      31      01E4    20$:    BRW      NXTSOB                    ; get next ifab
          50   53      D0      01E7    30$:    MOVL     R3, R0                    ; get saved error code
            F7 50      E8      01EA            BLBS     R0, 20$                   ; no problem if not
                               01ED                                               ;  write-accessed file
             FE5E      31      01ED            BRW      EXIT                      ; return error to caller
                               01F0    ERRBUG_BR:
             FEE8      31      01F0            BRW      ERRBUG                    ; extended branch
                               01F3
```

```
01F3  ;++
01F3  ; GETDEVIFID -- Get Device ID and File ID.
01F3  ;
01F3  ;  This routine returns the counted device name string,
01F3  ;  as well as the file id for the file open on the channel.
01F3  ;
01F3  ; Calling Sequence:
01F3  ;
01F3  ;       BSBW    GETDVIFID
01F3  ;
01F3  ; Input Parameters:
01F3  ;
01F3  ;       R9                IFAB address
01F3  ;       R3                address of 22-byte buffer to return device name string
01F3  ;       IFB$W_CHNL        channel #
01F3  ;
01F3  ; Implicit Inputs:
01F3  ;
01F3  ;       none
01F3  ;
01F3  ; Output Parameters:
01F3  ;
01F3  ;       R0,R1,R3          destroyed
01F3  ;
01F3  ; Implicit Outputs:
01F3  ;
01F3  ;       The counted ascii string for the device name is moved
01F3  ;       to the buffer provided, followed by the file id starting 16 bytes
01F3  ;       from the start of the buffer.
01F3  ;
01F3  ;--
01F3
```

```
                                01F3   GETDVIFID:
           0434 8F      BB      01F3           PUSHR    #^M<R2,R4,R5,R10>         ; Save regs.
                 53     DD      01F7           PUSHL    R3                        ; Save R3.
       5A    38 A9      D0      01F9           MOVL     IFB$L_FWA_PTR(R9),R10     ; Get FWA into R10.
   83      0190 CA      90      01FD           MOVB     FWA$Q_SHRFIL(R10),(R3)+ ; Move size of buffer id into first byte of
           0190 CA      28      0202           MOVC3    FWA$Q_SHRFIL(R10),-     ; Move device id name into buffer
           0194 DA              0206                    @FWA$Q_SHRFIL+4(R10),-
                 63             0209                    (R3)
                 53 8ED0        020A           POPL     R3                        ; Restore R3.
                                020D
                                020D
                                020D   ;
                                020D   ;  Now get the file ID from the FWA.
                                020D   ;  R3 = address of the specified output buffer
                                020D   ;
                                020D
             10 A3      D4      020D           CLRL     16(R3)                    ; Clear FID field in buffer.
             14 A3      B4      0210           CLRW     20(R3)
        07 69   05      E0      0213           BBS      #DEV$V_SQD,(R9),10$       ; branch if magtape (no FCB)
                06      28      0217           MOVC3    #6,-                      ; Move FID to buffer.
           01F8 CA              0219                    FWA$T_FIBBUF+FIB$W_FID(R10),-
             10 A3              021C                    16(R3)
           0434 8F      BA      021E   10$:    POPR     #^M<R2,R4,R5,R10>         ; Restore regs.
                        05      0222           RSB
                                0223
                                0223           .END
```

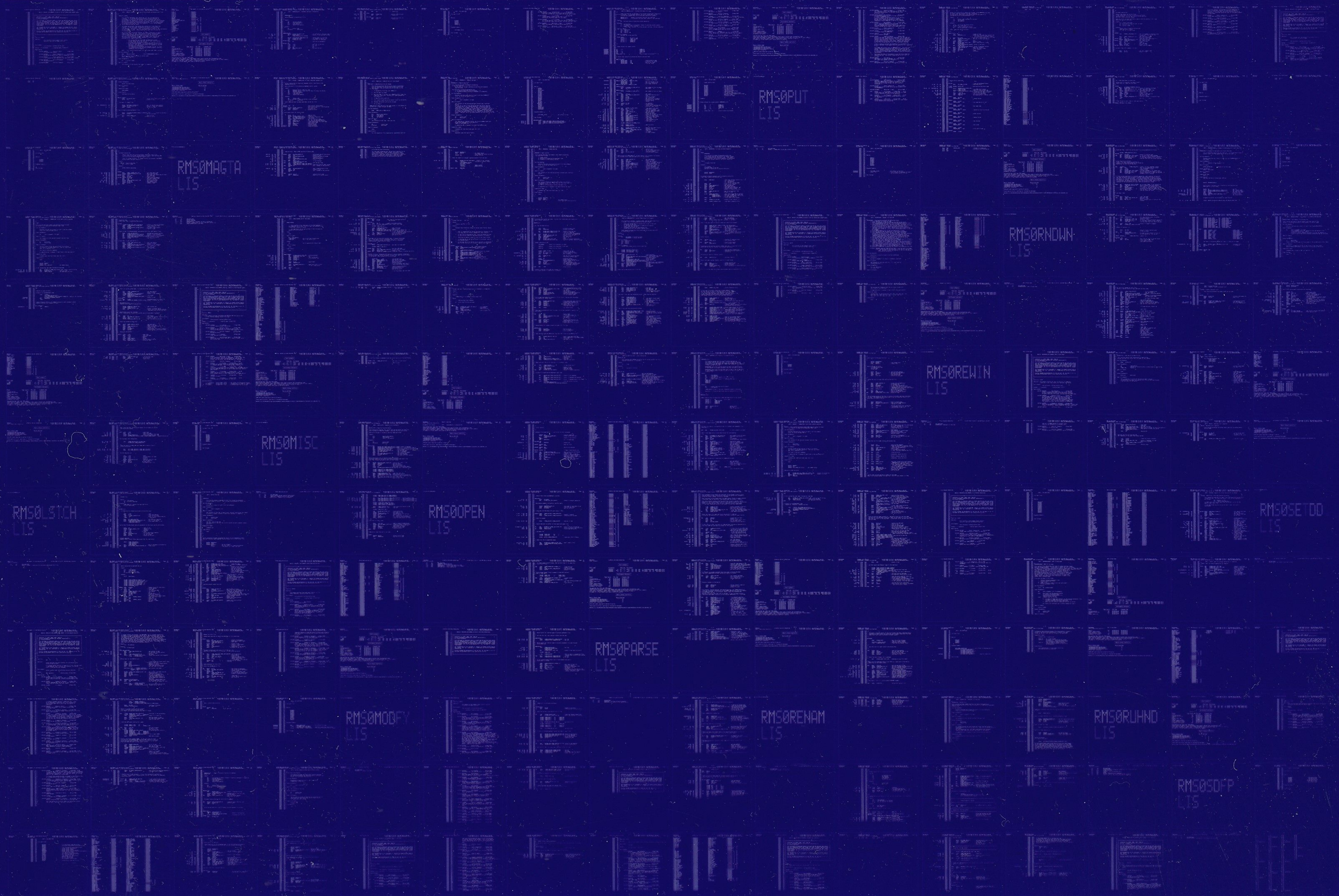
```
$$.PSECT_EP                      = 00000000                NWA$B_OSTYPE                       000000C4
$$RMSTEST                        = 0000001A                NWA$B_RFM                          000000C7
$$RMS_PBUGCHK                    = 00000010                NWA$B_RMS_RAC                      000000C8
$$RMS_TBUGCHK                    = 00000008                NWA$C_BLN                          00000800
$$RMS_UMODE                      = 00000004                NWA$K_BLN                          00000800
CANCEL                             0000014A R       01     NWA$L_ALLXABADR                    00000100
CHKIRB                             000000B8 R       01     NWA$L_DATXABADR                    00000104
CLOSE                              0000019B R       01     NWA$L_DEV                          000000C0
DEV$V_DIR                        = 00000003                NWA$L_FHCXABADR                    00000108
DEV$V_NET                        = 0000000D                NWA$L_KEYXABADR                    0000010C
DEV$V_RND                        = 0000001C                NWA$L_MSG_MASK                     000000D4
DEV$V_SQD                        = 00000005                NWA$L_PROXABADR                    00000110
ENBAST                             00000065 R       01     NWA$L_RDTXABADR                    00000114
ERRBUG                             000000DB R       01     NWA$L_SAVE_FLGS                    00000128
ERRBUG_BR                          000001F0 R       01     NWA$L_SUMXABADR                    00000118
ERRBUSY                            0000005B R       01     NWA$L_THREAD                       000000FC
EXIT                               0000004E R       01     NWA$L_XLTATTR                      00000238
FAB$C_BID                        = 00000003                NWA$L_XLTBUFFLG                    0000022C
FAB$C_BLN                        = 00000050                NWA$L_XLTCNT                       00000228
FAB$V_PPF_IND                    = 0000000E                NWA$L_XLTMAXINDX                   00000234
FAB$W_IFI                        = 00000002                NWA$L_XLTSIZ                       00000230
FIB$W_FID                        = 00000004                NWA$Q_ACS                          00000244
FTL$_IORNDN                      = FFFFFFEE                NWA$Q_BIGBUF                       00000170
FWA$Q_SHRFIL                     = 00000190                NWA$Q_BLD                          000000F0
FWA$T_FIBBUF                     = 000001F4                NWA$Q_FLG                          00000000
GETDVIFID                          000001F3 R       01     NWA$Q_INODE                        0000025C
IFB$L_FWA_PTR                    = 00000038                NWA$Q_IOSB                         000000D8
IFB$L_NWA_PTR                    = 0000003C                NWA$Q_LNODE                        00000160
IFB$L_PRIM_DEV                   = 00000000                NWA$Q_LOGNAME                      0000023C
IFB$L_SFSB_PTR                   = 00000078                NWA$Q_NCB                          00000264
IFB$V_ACCESSED                   = 00000025                NWA$Q_RCV                          000000E0
IFB$V_BUSY                       = 00000020                NWA$Q_SAVE_DESC                    00000120
IFB$V_RMS_STALL                  = 0000003A                NWA$Q_XLTBUF1                      0000024C
IFB$V_WRTACC                     = 00000030                NWA$Q_XLTBUF2                      00000254
IFB$W_CHNL                       = 00000020                NWA$Q_XMT                          000000E8
IMP$C_IOREFN                     = 0000001E                NWA$T_ACSBUF                       0000026C
IMP$L_IFABTBL                    = 00000018                NWA$T_AUXBUF                       000005E0
IMP$L_IRABTBL                    = 0000001C                NWA$T_DAP                          00000000
IMP$V_IORUNDOWN                  = 00000004                NWA$T_INODEBUF                     000004AC
IMP$W_ENTPERSEG                  = 00000020                NWA$T_ITM_ATTR                     00000200
IMP$W_NUM_IFABS                  = 00000022                NWA$T_ITM_END                      00000224
IMP$W_RMSSTATUS                  = 00000000                NWA$T_ITM_LST                      00000200
INHAST                             00000077 R       01     NWA$T_ITM_MAXINDX                  00000218
IRB$L_IRAB_LNK                   = 0000001C                NWA$T_ITM_STRING                   0000020C
IRB$V_BUSY                       = 00000020                NWA$T_NCBBUF                       0000052C
IRB$V_PPF_IMAGE                  = 00000022                NWA$T_NODEBUF                      00000169
IRB$V_RMS_STALL                  = 0000003A                NWA$T_RCVBUF                       000001A0
NOCANCEL                           00000155 R       01     NWA$T_SCAN                         00000100
NOERR                              00000198 R       01     NWA$T_TEMP                         00000120
NOERR1                             00000191 R       01     NWA$T_XLTBUF1                      000002AC
NWA$B_ALLXABCNT                    0000011C                NWA$T_XLTBUF2                      000003AC
NWA$B_DAP_RAC                      000000C9                NWA$T_XMTBUF                       000003C0
NWA$B_FILESYS                      000000C5                NWA$V_RCVAST                     = 00000004
NWA$B_KEYXABCNT                    0000011D                NWA$V_RCVQIO                     = 00000003
NWA$B_NETSTRSIZ                    0000016F                NWA$W_BUILD                        000000D2
NWA$B_NODBUFSIZ                    00000168                NWA$W_DAPBUFSIZ                    000000CA
NWA$B_ORG                          000000C6                NWA$W_DIR_OFF                      000000CC
```

```
NWA$W_DISPLAY                      000000D0
NWA$W_FIL_OFF                      000000CE
NWA$W_JNLXABJOP                    0000011E
NXTENT                             000000A7 R      01
NXTSEG                             000000A0 R      01
NXTSOB                             000000B3 R      01
PIO$A_TRACE                        ********   X    01
PIO$GW_IIOIMPA                     ********   X    01
PIO$GW_PIOIMPA                     ********   X    01
PIO$GW_STATUS                      ********   X    01
PIO$V_INHAST                     = 00000000
PSL$C_USER                       = 00000003
PSL$S_PRVMOD                     = 00000002
PSL$V_PRVMOD                     = 00000016
QUIET                              0000015A R      01
RDIFAB                             000000E2 R      01
RDIRAB                             00000121 R      01
RDNET                              0000010F R      01
RM$BUG                             ********   X    01
RM$LAST_CHANCE                     ********   X    01
RM$RU_UNLOCK                       ********   X    01
RM$UNLOCKALL                       ********   X    01
RMS$RMSRUNDWN                    = FFFFFFFE RG     01
RMS$_BUSY                        = 0001848C
RMS$_CCF                         = 0001C0DC
RMS$_IAL                         = 0001854C
RMSABORT                           00000055 R      01
RUNDWN                             00000096 R      01
SYS$CANCEL                         ********  GX    01
SYS$CLOSE                          ********   X    01
SYS$CLREF                          ********  GX    01
SYS$SETAST                         ********  GX    01
SYS$WAITFR                         ********  GX    01
TPT$L_RUNDWN                       ********   X    01
WAIT                               0000007F R      01
XITSUC                             0000004B 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                            00000223  (  547.)  01 (  1.)    PIC    USR    CON    REL    GBL NOSHR   EXE   RD   NOWRT NOVEC BYTE
$ABS$                             00000800  ( 2048.)  02 (  2.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE

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

Phase                       Page faults    CPU Time       Elapsed Time
-----                       -----------    --------       ------------
Initialization                      36     00:00:00.08    00:00:00.84
Command processing                 142     00:00:00.72    00:00:04.05
Pass 1                             417     00:00:16.06    00:00:41.85
Symbol table sort                    0     00:00:02.27    00:00:04.14
```

```
Pass 2                              109      00:00:02.95      00:00:06.37
Symbol table output                  19      00:00:00.18      00:00:00.26
Psect synopsis output                 1      00:00:00.04      00:00:00.04
Cross-reference output                0      00:00:00.00      00:00:00.00
Assembler run totals                726      00:00:22.31      00:00:57.63

The working set limit was 1650 pages
86675 bytes (170 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1649 non-local and 23 local symbols.
582 source lines were read in Pass 1, producing 14 object records in Pass 2.
37 pages of virtual memory were used to define 36 macros.

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

Macro library name                            Macros defined
------------------                            --------------

_$255$DUA28:[RMS.OBJ]RMS.MLB;1                          18
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                           3
_$255$DUA28:[SYSLIB]STARLET.MLB;2                       11
TOTALS (all libraries)                                  32

1838 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

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

0330 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



RMS0PUT
LIS

RMS0MAGTA
LIS

RMS0RNDWN
LIS

RMS0REWIN
LIS

RMS0MISC
LIS

RMS0LSTCH
LIS

RMS0OPEN
LIS

RMS0SETDD
LIS

RMS0PARSE
LIS

RMS0MODFY
LIS

RMS0RENAM
LIS

RMS0RUHND
LIS

RMS0SDFP
LIS