```
RRRRRRRRRRR      MMM          MMM      SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRRRRRRRRRR      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  SSSSSSSSSSS
RRR       RRR    MMM          MMM               SSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
```

```
RRRRRRRR    MM      MM    SSSSSSSS    000000    LL          SSSSSSSS  TTTTTTTTTT    CCCCCCCC  HH      HH
RRRRRRRR    MM      MM    SSSSSSSS    000000    LL          SSSSSSSS  TTTTTTTTTT    CCCCCCCC  HH      HH
RR      RR  MMMM  MMMM  SS          00      00  LL        SS            TT        CC          HH      HH
RR      RR  MMMM  MMMM  SS          00      00  LL        SS            TT        CC          HH      HH
RR      RR  MM  MM  MM  SS          00    0000  LL        SS            TT        CC          HH      HH
RR      RR  MM  MM  MM  SS          00    0000  LL        SS            TT        CC          HH      HH
RRRRRRRR    MM      MM    SSSSSS    00  00  00  LL          SSSSSS      TT        CC          HHHHHHHHHH
RRRRRRRR    MM      MM    SSSSSS    00  00  00  LL          SSSSSS      TT        CC          HHHHHHHHHH
RR  RR      MM      MM          SS  0000    00  LL                SS    TT        CC          HH      HH
RR  RR      MM      MM          SS  0000    00  LL                SS    TT        CC          HH      HH
RR    RR    MM      MM          SS  00      00  LL                SS    TT        CC          HH      HH
RR    RR    MM      MM          SS  00      00  LL                SS    TT        CC          HH      HH
RR      RR  MM      MM  SSSSSSSS    000000    LLLLLLLLLL  SSSSSSSS      TT          CCCCCCCC  HH      HH
RR      RR  MM      MM  SSSSSSSS    000000    LLLLLLLLLL  SSSSSSSS      TT          CCCCCCCC  HH      HH

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     1          $BEGIN  RMSOLSTCH,000,RM$RMS,<RMS ABORT I/O FOR PROCESS DELETE>
0000     2
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
0000    27 ;++
0000    28 ;
0000    29 ; Facility:
0000    30 ;                   RMS
0000    31 ;
0000    32 ; Abstract:
0000    33 ;                   Causes eof mark and dirty buffers for non-shared files
0000    34 ;                   to be written to disk.
0000    35 ;
0000    36 ;                   If a global buffer section is found for a file, the use count
0000    37 ;                   is decremented, and if we are last accessor, all buffer
0000    38 ;                   locks in the section are released.
0000    39 ;
0000    40 ; Environment:
0000    41 ;                   Kernel or exec mode, asynchronous process deletion.
0000    42 ;
0000    43 ; Author:
0000    44 ;                   Leo Laverdure           creation date: 21-Feb-1978
0000    45 ;
0000    46 ; Modified By:
0000    47 ;
0000    48 ;           V03-026 JEJ0035         J E Johnson             08-May-1984
0000    49 ;                   Another $GETLKIW problem: now look at the proper byte
0000    50 ;                   in the stack to find the currently granted mode of the
0000    51 ;                   lock.
0000    52 ;
0000    53 ;           V03-025 SHZ0015         Stephen H. Zalewski,    04-May-1984
0000    54 ;                   Routine rm$unmap_gbl now set up its own registers, so
0000    55 ;                   remove register setup from last chance.
0000    56 ;
0000    57 ;           V03-024 JEJ0024         J E Johnson             09-Apr-198
```

```
0000     58 ;               Fix broken argument pointer in $GETLKIW call.  Also
0000     59 ;               do not bugcheck if $GETLKIW fails with IVLOCKID as
0000     60 ;               the previous $DEQ had killed a new lock before it
0000     61 ;               completed.
0000     62 ;
0000     63 ;       V03-023 DGB0025         Donald G. Blair         09-Mar-1984
0000     64 ;               Allocate full-length fib to support access mode
0000     65 ;               protected files.
0000     66 ;
0000     67 ;       V03-022 SHZ0014         Stephen H. Zalewski,    22-Feb-1984
0000     68 ;               Do not do write the file header characteristics of a file
0000     69 ;               that is not accessed.
0000     70 ;
0000     71 ;       V03-021 SHZ0013         Stephen H. Zalewski     06-Dec-1983
0000     72 ;               Use FTL error code if $GETLKIW fails.  Also, do not attempt
0000     73 ;               to unmap a global buffer section if section is not accessible.
0000     74 ;               This could happen if process was in RMSOCLOSE, had unmapped
0000     75 ;               the section, but had not dequeued its lock on the global
0000     76 ;               section yet, and a $STOP was issued on the process.
0000     77 ;
0000     78 ;       V03-020 SHZ0012         S ephen H. Zalewski     10-Aug-1983
0000     79 ;               Change the way the GBSB lock is obtained.
0000     80 ;
0000     81 ;       V03-019 SHZ0011         Stephen H. Zalewski     04-Aug-1983
0000     82 ;               Fix broken branch.  Also make sure that the parent lock
0000     83 ;               is specified when taking out the global section lock.
0000     84 ;
0000     85 ;       V03-018 SHZ0010         Stephen H. Zalewski     02-Aug-1983
0000     86 ;               If global buffers are present, disassociate from global
0000     87 ;               section before lowering the EX lock on GBSB.  Also, do
0000     88 ;               not attempt to get EX lock on global buffer section if
0000     89 ;               we already have it.
0000     90 ;
0000     91 ;       V03-017 SHZ0009         Stephen H. Zalewski     28-Jul-1983
0000     92 ;               Add support for cluster global buffers.
0000     93 ;
0000     94 ;       V03-016 SHZ0008         Stephen H. Zalewski     26-Jun-1983
0000     95 ;               Fix bad branch.
0000     96 ;
0000     97 ;       V03-015 SHZ0007         Stephen H. Zalewski     22-Jun-1983
0000     98 ;               Fix broken instruction.
0000     99 ;
0000    100 ;       V03-014 SHZ0006         Stephen H. Zalewski     26-Apr-1983
0000    101 ;               Write out the file header and current bdb (if possible)
0000    102 ;               for non-shared files.
0000    103 ;
0000    104 ;       V03-013 JWH0137         Jeffrey W. Horn         24-Nov-1982
0000    105 ;               Make code pick up size of IFB/IRB table from
0000    106 ;               IMP$L_ENTPERSEG instead of IMP$C_ENTPERSEG.
0000    107 ;
0000    108 ;       V03-012 KBT0341         Keith B. Thompson       20-Sep-1982
0000    109 ;               Rewrite
0000    110 ;
0000    111 ;       V03-011 SHZ0005         Stephen H. Zalewski,    10-Sep-1982  22:05
0000    112 ;               Remove all references to SFD, SIFB and FRB structures as they
0000    113 ;               no longer exist.
0000    114 ;
```

```
0000    115 ;       V03-010 SHZ0004         Stephen H. Zalewski,    7-Sep-1982  22:28
0000    116 ;               If corrupt GBD found in global buffer section force a bugcheck
0000    117 ;               instead of simply fixing it.  Self-relative queue instructions
0000    118 ;               should prevent corruption.  This code that checks for corrupt
0000    119 ;               GBDs will be REMOVED at a future date.
0000    120 ;
0000    121 ;               Fix branch that caused process to attempt to deallocate a lock
0000    122 ;               on a global buffer section when it did not own a lock.
0000    123 ;
0000    124 ;       V03-009 SHZ0003         Stephen H. Zalewski,    6-Sep-1982  19:52
0000    125 ;               If stream has global buffers, dequeue lock on global buffer
0000    126 ;               section, and give back space used for GBSB.
0000    127 ;
0000    128 ;       V03-008 KBT0184         Keith B. Thompson       23-Aug-1982
0000    129 ;               Reorganize psects
0000    130 ;
0000    131 ;       V03-007 KBT0095         Keith B. Thompson       19-Jul-1982
0000    132 ;               Stuff IFAB into R9 when calling TAKE_SIFAB1
0000    133 ;
0000    134 ;       V03-006 TMK0001         Todd M. Kayz            02-Jul-1982
0000    135 ;               Deleted $NRPDEF. RMS cluster solution for next record
0000    136 ;               positioning eliminates the NRP list, and the corresponding
0000    137 ;               symbols.
0000    138 ;
0000    139 ;       V03-005 KDM0002         Kathleen D. Morse       28-Jun-1982
0000    140 ;               Added $PCBDEF.
0000    141 ;
0000    142 ;       V03-004 SHZ0002         Stephen H. Zalewski,    11-Jun-1982  12:33
0000    143 ;               Fix misspelling of SFD$B_FRBFLGS.
0000    144 ;
0000    145 ;       V03-003 SHZ0001         Stephen H. Zalewski,    8-Jun-1982  16:32
0000    146 ;               Before releasing SIFB, clear the PID in the temp FRB.
0000    147 ;
0000    148 ;               Set flag saying are in kernel mode just before we release
0000    149 ;               SFD so that if we stall, we stall correctly.
0000    150 ;
0000    151 ;--
0000    152 ;
0000    153
```

```
0000   155          .SBTTL  DECLARATIONS
0000   156
0000   157 ;
0000   158 ; Include Files:
0000   159 ;
0000   160
0000   161          $ATRDEF                         ; File attribute code defintions
0000   162          $BDBDEF                         ; rms buffer descriptor block definitions
0000   163          $DEVDEF                         ;
0000   164          $FIBDEF                         ; file information block definitions
0000   165          $GBDDEF                         ; glcbal buffer descriptors
0000   166          $GBHDEF                         ; global buffer header
0000   167          $GBPBDEF                        ; global buffer pointer blocks
0000   168          $GBSBDEF                        ; Global buffer synchronization block defini
0000   169          $IFBDEF                         ; rms ifab definitions
0000   170          $IMPDEF                         ; rms impure area definitions
0000   171          $IODEF                          ; qio function code definitions
0000   172          $IRBDEF                         ; rms irab definitions
0000   173          $LCKDEF                         ; Lock manger call definitions
0000   174          $LKIDEF                         ; $GETLKI definitions
0000   175          $PSLDEF                         ; psl bit definitions
0000   176          $QIODEF                         ; qio parameter definitions
0000   177          $RMSDEF                         ;
0000   178          $SSDEF                          ; sytem status code definitions
0000   179
```

RMS
Sym
$$.
$$A
$$R
$$R
$$R
$$R
$$T
ATR
BDB
BDB
BDB
BDB
BDB
BDB
BDB
BDB
BDB
BDB
BDB
CHE
CHK
CLN
CLO
CLO
DEQ
DEV
DOQ
EXE
EXI
EXT
FIB
FIB
FIB
FIB
FIB
FIB
FTL
FTL
GBD
GBD
GBD
GBD
GBD
GBD
GBH
GBH
GBH
GBH
GBP
GBP
GBP
GBP
GBP
GBP
GBP
GBS

```
0000    181          .SBTTL  RM$LAST_CHANCE - Clean up write accessed non-shared disk files
0000    182
0000    183 ;++
0000    184 ;
0000    185 ;  RMS$LAST_CHANCE      -
0000    186 ;
0000    187 ;       this routine implements the "abort rms i/o" rms run down option
0000    188 ;       (type=2).  it is called during asynchronous process deletion to
0000    189 ;       allow RMS32 to write out the contents of dirty buffers and record
0000    190 ;       the eof mark for sequential file org. disk files that are write accessed.
0000    191 ;       because this routine's being called from kernel mode ast level violates
0000    192 ;       rms synchronization and interlocking mechanisms, it is not guaranteed
0000    193 ;       to succeed in writing out the file correctly.  it will do as good a
0000    194 ;       job as possible and not fault.
0000    195 ;
0000    196 ;       the image and process ifab tables are scanned for files requiring
0000    197 ;       cleanup.  if any are found the appropriate buffer writing and or
0000    198 ;       file closing is performed.  there is no attempt to clean up the
0000    199 ;       rms data base.  in fact, the only rms impure data written is the rearranging
0000    200 ;       of the file attributes.  there should be no more calls to rms for
0000    201 ;       this process.
0000    202 ;
0000    203 ;       any errors which occur are ignored, as there is no place to report
0000    204 ;       them in any case.  in fact, this routine is almost paranoid about
0000    205 ;       checking the validity and accessibility of the rms structures.
0000    206 ;       this is required because this code is entered via kernel ast, which
0000    207 ;       may have interrupted rms running in exec mode, thus leaving the
0000    208 ;       rms structures in unknown states. because of this, a number of
0000    209 ;       problems exist, some of which are:
0000    210 ;
0000    211 ;               1. the current vbn may have been decremented already
0000    212 ;                  in the "buffer dirty but not valid" case, thus causing
0000    213 ;                  one block too few to be written.
0000    214 ;               2. a record being updated in a buffer may only be partially
0000    215 ;                  moved into that buffer, thus getting a mixture of old and
0000    216 ;                  new data.  the same problem exists for updates on records
0000    217 ;                  spanning block boundaries.
0000    218 ;               3. i/o completion on mailboxes and magtape, as well as other
0000    219 ;                  file organizations may not be finished properly, possibly
0000    220 ;                  leading to file corruption.
0000    221 ;
0000    222 ;       NOTE:   This routine should never call any normal RMS (RM$) routines.
0000    223 ;
0000    224 ; Calling sequence:
0000    225 ;
0000    226 ;       BRW     RM$LAST_CHANCE          ; ( from SYS$RMSRUNDWN (arg2=2) )
0000    227 ;
0000    228 ;       May be called from either exec or kernel mode.
0000    229 ;       Does a RET when done.
0000    230 ;
0000    231 ; Input Parameters:
0000    232 ;
0000    233 ;       R7      caller's mode
0000    234 ;
0000    235 ; Implicit Inputs:
0000    236 ;
0000    237 ;       the contents of the process and image i/o segments
```

```
                                0000   238 ;
                                0000   239 ; Output Parameters:
                                0000   240 ;
                                0000   241 ;       R0-R11 are destroyed
                                0000   242 ;
                                0000   243 ; Implicit Outputs:
                                0000   244 ;       none
                                0000   245 ;
                                0000   246 ; Completion Codes:
                                0000   247 ;
                                0000   248 ;       RMS$_NORMAL
                                0000   249 ;
                                0000   250 ; Side Effects:
                                0000   251 ;
                                0000   252 ;       see description above.
                                0000   253 ;
                                0000   254 ;--
                                0000   255
                                0000   256 RMS$LAST_CHANCE::
                01    57    91  0000   257         CMPB    R7,#PSL$C_EXEC                  ; caller sufficiently privileged?
                      16    1A  0003   258         BGTRU   EXIT                            ; branch if not
                                0005   259
                                0005   260 ;
                                0005   261 ;  run down the image files
                                0005   262 ;
                                0005   263
          5B    00000000'9F   DE  0005   264         MOVAL   @#PIO$GW_IIOIMPA,R11           ; get iio impure area address
                                000C   265         IFNORD  #1,(R11),EXIT,R7                ; branch if page not readable
                      0F    10  0012   266         BSBB    RUNDWN                          ; do the run down
                                0014   267
                                0014   268 ;
                                0014   269 ; now run down process-permanent files
                                0014   270 ;
                                0014   271
             5B    0000'CB    DE  0014   272         MOVAL   W^PIO$GW_PIOIMPA-PIO$GW_IIOIMPA(R11),R11        ; point to process i
                         08    10  0019   273         BSBB    RUNDWN                          ; do the run down
                                001B   274
          50    00010001 8F   D0  001B   275 EXIT:   MOVL    #RMS$_NORMAL,R0                 ; show success
                         04    0022   276         RET                                     ; back to caller
                               0023   277
```

```
                     0023   279
                     0023   280 ;++
                     0023   281 ;
                     0023   282 ; run down subroutine:
                     0023   283 ;
                     0023   284 ;       Checks ifab table for write-accessed, nonshared, disk files.
                     0023   285 ;       If any found, write the current buffer, if dirty, and then deaccess
                     0023   286 ;       the file to write the current eof mark to disk.
                     0023   287 ;
                     0023   288 ;       Check for global buffers.  If global buffers are present, decrement
                     0023   289 ;       access counts and mark invalid.
                     0023   290 ;
                     0023   291 ; inputs:
                     0023   292 ;
                     0023   293 ;       R11 - impure area addr
                     0023   294 ;       R7 - caller's mode
                     0023   295 ;
                     0023   296 ; outputs:
                     0023   297 ;
                     0023   298 ;       R0-R6, R8-R10 destroyed
                     0023   299 ;
                     0023   300 ;--
                     0023   301
                     0023   302 RUNDWN:
55   18 AB   D0      0023   303         MOVL    IMP$L_IFABTBL(R11),R5           ; get ifab table addr
58   20 AB   3C      0027   304         MOVZWL  IMP$W_ENTPERSEG(R11),R8         ; get # entries/seg
        85   DD      002B   305 NXTSEG: PUSHL   (R5)+                           ; save addr next table seg on stack
   5A   85   D0      002D   306 NXTENT: MOVL    (R5)+,R10                       ; get ifab addr
        17   12      0030   307         BNEQ    RDIFAB                          ; branch if one
     F8 58   F5      0032   308 NXTSOB: SOBGTR  R8,NXTENT                       ; keep scanning segment
      55   8E   D0   0035   309         MOVL    (SP)+,R5                        ; get next segment addr
           0E   13   0038   310         BEQL    10$                             ; branch if no more
   58   20 AB   3C   003A   311         MOVZWL  IMP$W_ENTPERSEG(R11),R8         ; get # entries/seg
56 58   04   C5   003E   312         MULL3   #4,R8,R6                        ; get segment size
65 56   57   0C   0042   313         PROBER  R7,R6,(R5)                      ; segment readable?
        E3   12   0046   314         BNEQ    NXTSEG                          ; branch if yes
             05   0048   315 10$:    RSB
                  0049   316
```

```
                     0049   318 ;
                     0049   319 ; Found an ifab.
                     0049   320 ; Check that ifab is accessible and valid.
                     0049   321 ;
                     0049   322
                     0049   323 RDIFAB:
                     0049   324         IFNOWRT #IFB$C_BLN,(R10),NXTSOB,R7       ; branch if ifab not writeable
0B   08 AA   91      0051   325         CMPB    IFB$B_BID(R10),#IFB$C_BID        ; is it really an ifab?
        DB   12      0055   326         BNEQ    NXTSOB                           ; ignore if not
2E   09 AA   91      0057   327         CMPB    IFB$B_BLN(R10),#IFB$C_BLN/4      ; at least right length?
        D5   1F      005B   328         BLSSU   NXTSOB                           ; ignore if bad
                     005D   329
                     005D   330 ;
                     005D   331 ; Check to see if write-accessed, non-shared disk file, and ignore if not.
                     005D   332 ;
      014C   30      005D   333         BSBW    CHECK_GBL_BUFFERS                ; Release any global buffers.
CE 6A   25   E1      0060   334         BBC     #IFB$V_ACCESSED,(R10),NXTSOB     ; Branch if not accessed.
CA 6A   30   E1      0064   335         BBC     #IFB$V_WRTACC,(R10),NXTSOB       ; Branch if not write accessed.
C6 6A   1C   E1      0068   336         BBC     #DEV$V_RND,IFB$L_PRIM_DEV(R10),NXTSOB ; Branch if not disk.
     78 AA   D5      006C   337         TSTL    IFB$L_SFSB_PTR(R10)              ; Is this file shared?
        C1   12      006F   338         BNEQ    NXTSOB                           ; Yes, skip it.
                     0071   339
                     0071   340 ;
                     0071   341 ; Find irab, if one, and verify it.
                     0071   342 ;
59   1C AA   D0      0071   343         MOVL    IFB$L_IRAB_LNK(R10),R9           ; Get irab address
        2A   13      0075   344         BEQL    CLOSE_BR                         ; Branch if none.
                     0077   345         IFNORD  #IRB$C_BLN_SEQ,(R9),NXTSOB,R7    ; branch if not readable.
0A   08 A9   91      007F   346         CMPB    IRB$B_BID(R9),#IRB$C_BID         ; Is is really an irab?
        AD   12      0083   347         BNEQ    NXTSOB                           ; No, skip file.
1B   09 A9   91      0085   348         CMPB    IRB$B_BLN(R9),#IRB$C_BLN_SEQ/4   ; long enough?
        A7   1F      0089   349         BLSSU   NXTSOB                           ; No, skip file.
                     008B   350
```

```
                     008B   352 ;
                     008B   353 ;  check for valid current bdb.  if found and dirty, write it.
                     008B   354 ;
                     008B   355
    54   20 A9   D0  008B   356          MOVL    IRB$L_CURBDB(R9),R4             ; get bdb address
           10   13  008F   357          BEQL    CLOSE_BR                        ; brach if none
                    0091   358          IFNORD  #BDB$C_BLN,(R4),CLOSE_BR,R7     ; branch if not readable
                    0099   359          ASSUME  BDB$B_BLN EQ BDB$B_BID+1
        08 A4   B1  0099   360          CMPW    BDB$B_BID(R4),-                 ; valid bdb?
      140C 8F       009C   361                  #BDB$C_BID+<<BDB$C_BLN/4>@8>
           03   13  009F   362          BEQL    CHKDRT                          ; branch if yes
                    00A1   363 CLOSE_BR:
         004C   31  00A1   364          BRW     CLOSE                           ; extended branch to close file
47 0A A4    01   E1  00A4   365 CHKDRT: BBC     #BDB$V_DRT,BDB$B_FLGS(R4),CLOSE ; branch if buffer not dirty
                    00A9   366
                     00A9   367 ;
                     00A9   368 ;  bdb marked dirty.
                     00A9   369 ;  compute size of buffer to be written.
                     00A9   370 ;
                     00A9   371
    56   48 A4   9A  00A9   372          MOVZBL  BDB$B_REL_VBN(R4),R6            ; get current vbn
                    00AD   373          ASSUME  BDB$V_VAL EQ 0
      02 0A A4   E8  00AD   374          BLBS    BDB$B_FLGS(R4),10$              ; branch if buffer valid
           56   D7  00B1   375          DECL    R6                              ; buffer dirty but marked invalid,
                    00B3   376                                                  ;  decrement the current vbn data.
                     00B3   377 ;
                     00B3   378 ; compute byte count for transfer
                     00B3   379 ; (i.e., transfer all blocks thru current vbn or # val_vbns if greater)
                     00B3   380 ;
                     00B3   381
           56   B6  00B3   382 10$:    INCW    R6                              ; get # vbns
        49 A4   56   91  00B5   383          CMPB    R6,BDB$B_VAL_VBNS(R4)           ; is current greater than  # valid v
           04   1E  00B9   384          BGEQU   20$                             ; branch if yes
        56   49 A4   9A  00BB   385          MOVZBL  BDB$B_VAL_VBNS(R4),R6           ; no - so use # valid vbns
                    00BF   386
                     00BF   387 ;
                     00BF   388 ;  check for extend of disk file
                     00BF   389 ;
                     00BF   390
 51    56   1C A4   C1  00BF   391 20$:    ADDL3   BDB$L_VBN(R4),R6,R1             ; get end vbn+1
           51   D7  00C4   392          DECL    R1                              ; make it end vbn
        51   70 AA   C2  00C6   393          SUBL2   IFB$L_HBK(R10),R1               ; need to extend file?
           09   15  00CA   394          BLEQ    WRITE                           ; branch if not
         0059   30  00CC   395          BSBW    EXTEND_FILE                     ; Extend the file
        03 50   E8  00CF   396          BLBS    R0,WRITE                        ; Write buffer if extend succeeded.
         FF5D   31  00D2   397          BRW     NXTSOB                          ; Extend failed, go to next file.
                    00D5   398 ;
                     00D5   399 ;  now write the buffer
                     00D5   400 ;
                     00D5   401
           7E   7C  00D5   402 WRITE:  CLRQ    -(SP)                   ; make iosb
      52   5E   D0  00D7   403          MOVL    SP,R2                   ; save iosb addr
           7E   7C  00DA   404          CLRQ    -(SP)                   ; p6=p5=0
           7E   D4  00DC   405          CLRL    -(SP)                   ; p4=0
        1C A4   DD  00DE   406          PUSHL   BDB$L_VBN(R4)           ; p3=vbn
 7E    56   09   9C  00E1   407          ROTL    #9,R6,-(SP)             ; p2=# of bytes to write
        18 A4   DD  00E5   408          PUSHL   BDB$L_ADDR(R4)          ; p1=buffer addr
```

```
                50    30    D0  00E8   409          MOVL    #IO$_WRITEVBLK,R0       ; i/o function code
                   0087    30  00EB   410          BSBW    DOQIO                   ; do the write
                     03    BA  00EE   411          POPR    #^M<R0,R1>              ; clean stack
                               00F0   412
```

```
                                00F0   414 ;++
                                00F0   415 ;
                                00F0   416 ;  Now close the file, building attribute list on stack to rewrite
                                00F0   417 ;  the record attributes.
                                00F0   418 ;
                                00F0   419 ;--
                                00F0   420
                          7E    7C 00F0   421 CLOSE:  CLRQ    -(SP)                       ; make iosb
                        52  5E    D0 00F2   422         MOVL    SP,R2                       ; save iosb addr
                            00    DD 00F5   423         PUSHL   #0                          ; end of attribute list flag
                          50 AA   DF 00F7   424         PUSHAL  IFB$B_RFMORG(R10)           ; write attributes from ifab
                00040016 8F       DD 00FA   425         PUSHL   #<ATR$C_RECATTR@16>+<IFB$C_FHAEND-IFB$B_RFMORG> ; length & record at
                                0100   426
                                0100   427 ;
                                0100   428 ; Put org back into rfmorg byte.
                                0100   429 ;
                                0100   430
                    04  23 AA   F0 0100   431         INSV    IFB$B_ORGCASE(R10),#IFB$V_ORG,-
                    50 AA   04    0104   432                 #IFB$S_ORG,IFB$B_RFMORG(R10)
                                0107   433
                                0107   434 ;
                                0107   435 ;  swap the words of IFB$L_HBK and IFB$L_EBK to match files-11
                                0107   436 ;  on-disk structure
                                0107   437 ;
                                0107   438
54 AA   70 AA   10   9C 0107   439         ROTL    #16,IFB$L_HBK(R10),IFB$L_HBK_DISK(R10)
58 AA   74 AA   10   9C 010D   440         ROTL    #16,IFB$L_EBK(R10),IFB$L_EBK_DISK(R10)
                                0113   441
                                0113   442 ;
                                0113   443 ;  do the deaccess qio
                                0113   444 ;
                                0113   445
                          7E    D4 0113   446         CLRL    -(SP)                       ; p6=0
                        04 AE   DF 0115   447         PUSHAL  4(SP)                       ; p5=attribute list addr
                          7E    7C 0118   448         CLRQ    -(SP)                       ; p4=p3=0
                          7E    7C 011A   449         CLRQ    -(SP)                       ; p2=p1=0
                    50    34    9A 011C   450         MOVZBL  #IO$_DEACCESS,R0            ; deaccess function code
                       0053   30 011F   451         BSBW    DOQIO                       ; do the deaccess
                    5E    14    C0 0122   452         ADDL2   #20,SP                      ; clean stack
                       FF0A   31 0125   453         BRW     NXTSOB                      ; go handle next file
                                0128   454
```

```
                     0128   456 ;++
                     0128   457 ;
                     0128   458 ;  extend file
                     0128   459 ;
                     0128   460 ;
                     0128   461 ; Inputs:
                     0128   462 ;       r10 - IFB address
                     0128   463 ;       r1  - # blocks required
                     0128   464 ;
                     0128   465 ; Outputs:
                     0128   466 ;       IFB$L_HBK will contain the new high block.
                     0128   467 ;       r0,r2,r3 destroyed
                     0128   468 ;
                     0128   469 ;--
                     0128   470
                     0128   471 EXTEND_FILE:
                     0128   472 ;
                     0128   473 ;  initialize an io status block and a fib on the stack
                     0128   474 ;
                     0128   475
          7E    7C   0128   476         CLRQ    -(SP)                           ; iosb
     52   5E    D0   012A   477         MOVL    SP,R2                           ; save iosb address
                     012D   478         .REPT   <<FIB$C_LENGTH+7>/8>            ; clear fib on stack
                     012D   479         CLRQ    -(SP)                           ; note: 8 clrg's faster/smaller
          7E    7C   012D   480         .ENDM                                   ;   than movc5
                     013D   481
  18 AE   51    D0   013D   482         MOVL    R1,FIB$L_EXSZ(SP)               ; set # blocks required
16 AE   80 8F   90   0141   483         MOVB    #FIB$M_EXTEND,FIB$W_EXCTL(SP)   ; say it's an extend
     4F AA    90   0146   484         MOVB    IFB$B_AGENT_MODE(R10),-         ; store agent mode in fib
     2E AE        0149   485                 FIB$B_AGENT_MODE(SP)
                     014B   486
                     014B   487 ;
                     014B   488 ; build the fib descriptor and qio parameters on the stack and do the extend.
                     014B   489 ;
                     014B   490
        5E    DD   014B   491         PUSHL   SP                              ; addr of fib
     7E 40 8F    9A   014D   492         MOVZBL  #FIB$C_LENGTH,-(SP)             ; length of fib
        7E    7C   0151   493         CLRQ    -(SP)                           ; p6=p5=0
        7E    7C   0153   494         CLRQ    -(SP)                           ; p4=p3=0
        7E    D4   0155   495         CLRL    -(SP)                           ; p2=0
     14 AE    DF   0157   496         PUSHAL  20(SP)                          ; p1=fib descriptor address
  50    36    D0   015A   497         MOVL    #IO$_MODIFY,R0                  ; io function code
    0015      30   015D   498         BSBW    DOQIO                           ; do the extend
   0A 50      E9   0160   499         BLBC    R0,30$                          ; branch on failure
                     0163   500
                     0163   501 ;
                     0163   502 ; extend complete.
                     0163   503 ; update ifab hi block field, deallocate the fib, and return
                     0163   504 ;
                     0163   505
     20 AE    C1   0163   506         ADDL3   FIB$L_EXSZ+8(SP),-
     24 AE        0166   507                 FIB$L_EXVBN+8(SP),-
     70 AA        0168   508                 IFB$L_HBK(R10)
     70 AA    D7   016A   509         DECL    IFB$L_HBK(R10)                  ; get # of highest allocated blk
5E 00000050 8F    C0   016D   510 30$:    ADDL2   #16+<^FIB$C_LENGTH+7>&^XF8>,SP  ; clean stack
                  05   0174   511         RSB
                     0175   512
```

```
                     0175   514 ;++
                     0175   515 ;
                     0175   516 ;  doqio subroutine to perform the qio
                     0175   517 ;
                     0175   518 ;  inputs:
                     0175   519 ;
                     0175   520 ;       R10     ifab address
                     0175   521 ;       R2      iosb address
                     0175   522 ;       R0      qio function code
                     0175   523 ;       P1 thru P6 already on stack.
                     0175   524 ;
                     0175   525 ;  outputs:
                     0175   526 ;
                     0175   527 ;       R0,R1   destroyed
                     0175   528 ;
                     0175   529 ;--
                     0175   530
                     0175   531          ASSUME  QIO$_ASTPRM EQ <QIO$_P1 - 4>
                     0175   532          ASSUME  QIO$_ASTADR EQ <QIO$_ASTPRM - 4>
                     0175   533          ASSUME  QIO$_IOSB EQ <QIO$_ASTADR - 4>
                     0175   534          ASSUME  QIO$_FUNC EQ <QIO$_IOSB - 4>
                     0175   535          ASSUME  QIO$_CHAN EQ <QIO$_FUNC - 4>
                     0175   536          ASSUME  QIO$_EFN EQ <QIO$_CHAN - 4>
                     0175   537          ASSUME  QIO$_NARGS EQ 12
                     0175   538
            08   BA  0175   539 DOQIO:   POPR    #^M<R3>                         ; Remove return PC from stack.
            7E   7C  0177   540          CLRQ    -(SP)                           ; no ast wanted
            52   DD  0179   541          PUSHL   R2                              ; iosb addr
            50   DD  017B   542          PUSHL   R0                              ; i/o function code
      7E    20 AA    3C  017D   543          MOVZWL  IFB$W_CHNL(R10),-(SP)           ; i/o channel
            1E   DD  0181   544          PUSHL   #IMP$C_IOREFN                   ; efn
00000000'9F    0C   FB  0183   545          CALLS   #12,@#SYS$QIO                   ; do the qio
         1D 50   E9  018A   546          BLBC    R0,20$                          ; branch on failure
      50    62   D0  018D   547 10$:     MOVL    (R2),R0                         ; get status from iosb
            18   12  0190   548          BNEQ    20$                             ; and branch if io really done
                     0192   549          $CLREF_S        #IMP$C_IOREFN           ; clear qio event flag
            62   D5  019B   550          TSTL    (R2)                            ; done now?
            EE   12  019D   551          BNEQ    10$                             ; branch if yes
                     019F   552          $WAITFR_S       #IMP$C_IOREFN           ; wait for flag
            E3   11  01A8   553          BRB     10$                             ; go check if done
            63   17  01AA   554 20$:     JMP     (R3)                            ; Return.
                     01AC   555
```

```
                     01AC    557 ;++
                     01AC    558 ;
                     01AC    559 ;  If global buffers are present, the BDB/GBPB list is scanned looking for
                     01AC    560 ;  any GBPB's currently accessed (usecnt neq 0) and the access count in the
                     01AC    561 ;  corresponding GBD is decremented for those found.  This does NOT guarantee
                     01AC    562 ;  that GBD's will always be correct, but the error is that one might not
                     01AC    563 ;  be decremented when it should, rather than decrementing one incorrectly.
                     01AC    564 ;  Because of the kernel ast, extraordinary measures are necessary to guarantee
                     01AC    565 ;  correctness, and the failure to decrement the GBD use count simply makes
                     01AC    566 ;  that buffer ineligible for replacement from the cache.  The window is fairly
                     01AC    567 ;  small and this is considered acceptable behavior.
                     01AC    568 ;
                     01AC    569 ; Inputs:
                     01AC    570 ;       r10 - ifb address
                     01AC    571 ;
                     01AC    572 ; Outputs:
                     01AC    573 ;       r0 - r9 destroyed
                     01AC    574 ;
                     01AC    575 ;--
                     01AC    576 CHECK_GBL_BUFFERS:
          56  0088 CA   D0  01AC    577         MOVL    IFB$L_GBH_PTR(R10),R6           ; get pointer to gbh
                  01   12  01B1    578         BNEQ    10$                             ; if one then chk gbl buffers
                       05  01B3    579         RSB                                     ; Return if none.
            54  7C AA   D0  01B4    580 10$:    MOVL    IFB$L_GBSB_PTR(R10),R4          ; get gbsb
                     01B8    581         $DEQ_S  LKID    = GBSB$L_LOCK_ID(R4),-  ; Cancel any current lock requests
                     01B8    582                 FLAGS   = #LCK$M_CANCEL
                     01C8    583
                     01C8    584 ;
                     01C8    585 ; Build item list for $GETLKI call.
                     01C8    586 ;
                     01C8    587
                  7E 7C  01C8    588         CLRQ    -(SP)                           ; 4 byte buffer & 4 byte return leng
                  7E D4  01CA    589         CLRL    -(SP)                           ; End of item list.
               08 AE DF  01CC    590         PUSHAL  8(SP)                           ; Address of return length buffer.
               08 AE DF  01CF    591         PUSHAL  8(SP)                           ; Address of buffer
         01010003 8F DD  01D2    592         PUSHL   #<LKI$_STATE@16>+3              ; Item code and buffer size.
             50   5E D0  01D8    593         MOVL    SP,R0                           ; Get the item list pointer
                     01DB    594
                     01DB    595         $GETLKIW_S EFN      = #IMP$C_ASYQIOEFN,- ; Get current lock mode.
                     01DB    596                    LKIDADR = GBSB$L_LOCK_ID(R4),-
                     01DB    597                    ITMLST  = (R0)
                     01EF    598
            52  11 AE   9A  01EF    599         MOVZBL  LKI$B_STATE_GRMODE+16(SP), R2   ; Get current lock mode into R2.
                5E 18   C0  01F3    600         ADDL2   #24,SP                          ; Clean up stack.
                  0F 50 E8  01F6    601         BLBS    R0,15$                          ; Continue is successful.
          2124 8F   50 B1   01F9    602         CMPW    R0,#SS$_IVLOCKID                ; The only acceptable error
                  01   12  01FE    603         BNEQ    12$                             ; (caused by $DEQ above), is
                       05  0200    604         RSB                                     ; from last chance coming in
                           0201    605                                                 ; before $ENQ had created the
                           0201    606                                                 ; new lock.
                           0201    607 12$:    RMSPBUG FTL$_GETLKIFAIL                 ; Otherwise, SHOULD NEVER FAIL.
                52  05   91  0208    608 15$:    CMPB    #LCK$K_EXMODE, R2               ; Do we already have EX lock?
                      28 13  020B    609         BEQL    20$                             ; Yes, then do not queue for lock.
                           020D    610
                           020D    611         $ENQW_S EFN     = #IMP$C_ASYQIOEFN,-
                           020D    612                 LKMODE  = #LCK$K_EXMODE,-       ; get exclusive access to the
                           020D    613                 PARID   = G^EXE$GL_SYSID_LOCK,- ;  global buffer section
```

```
                     020D   614                LKSB    = GBSB$L_LKSB(R4),-
                     020D   615                RESNAM  = GBSB$Q_FILENAME(R4),-
                     020D   616                FLAGS   = #LCK$M_SYSTEM!LCK$M_SYNCSTS!LCK$M_VALBLK
                     022B   617
          07 50  E8  022B   618        BLBS    R0,20$                          ; Continue if success.
                     022E   619        RMSPBUG FTL$_ENQDEQFAIL                 ; WE CANNOT FAIL!
                     0235   620
                     0235   621 ;
                     0235   622 ; There is a global buffer section present.
                     0235   623 ; Check if section accessible and valid.
                     0235   624 ; Scan GBPB's and deaccess appropriate GBD's if found.
                     0235   625 ;
                     0235   626
                     0235   627 20$.    IFNORD  #GBH$C_BLN,(R6),DEQLCK,R7       ; Skip over if not accessible.
                     023D   628         ASSUME  <GBH$B_BID + 1> EQ      GBH$B_BLN
        1611 8F      B1    023D   629         CMPW    #<GBH$C_BID+<GBH$C_BLN/4@8>>,-  ; check if it looks ok.
          08 A6            0241   630                 GBH$B_BID(R6)
             4B      12    0243   631         BNEQ    DEQLCK                  ; NEQ the header is bad.
    50     40 AA   DE    0245   632         MOVAL   IFB$L_BDB_FLNK(R10), R0         ; Get BDB queue header addr.
        51   50    D0    0249   633         MOVL    R0,R1                   ; Save for end test.
                     024C   634
                     024C   635 ;
                     024C   636 ;  Scan the bdb queue looking for a gbpb
                     024C   637 ;
                     024C   638
                     024C   639         ASSUME  BDB$L_FLINK     EQ      0
                     024C   640         ASSUME  BDB$L_FLINK     EQ      GBPB$L_FLINK
                     024C   641
        50   60      D0    024C   642 SCAN:   MOVL    (R0),R0                 ; Get next element.
        51   50      D1    024F   643         CMPL    R0,R1                   ; At end of list?
             30      13    0252   644         BEQL    CLN_UP                  ; EQL then at end.
                     0254   645         IFNORD  #GBPB$C_BLN,(R0),SCAN,R7        ; If not accessible, skip on.
                     025A   646         ASSUME  GBPB$B_BID      EQ      BDB$B_BID
                     025A   647         ASSUME  GBPB$B_BLN      EQ      BDB$B_BLN
                     025A   648         ASSUME  <GBPB$B_BID + 1>EQ      GBPB$B_BLN
        0A15 8F      B1    025A   649         CMPW    #<GBPB$C_BID+<GBPB$C_BLN/4@8>>,- ; Is it legit GBPB?
          08 A0            025E   650                 GBPB$B_BID(R0)
             EA      12    0260   651         BNEQ    SCAN                    ; NEQ then no good.
                     0262   652
                     0262   653 ;
                     0262   654 ; Found a valid and readable GBPB.  If the use count not zero,
                     0262   655 ; fix count on corresponding GBD and mark GBD invalid in case the
                     0262   656 ; buffer was partially modified.  This will force it to be read
                     0262   657 ; from disk when the next accessor gains control.
                     0262   658 ;
                     0262   659
          0C A0      B5    0262   660         TSTW    GBPB$L_USERS(R0)                ; is it in use?
             E5      13    0265   661         BEQL    SCAN                    ; EQL no, so cont scan
    52     24 A0      D0    0267   662         MOVL    GBPB$L_GBD_PTR(R0),R2           ; get addr of GBD
                     026B   663         IFNOWRT #GBD$C_BLN,(R2),SCAN,R7         ; if not writeable, cont scan
                     0271   664         ASSUME  <GBD$B_BID+1>   EQ      GBD$B_BLN
        0A13 8F      B1    0271   665         CMPW    #GBD$C_BID+<GBD$C_BLN/4@8>,-    ; is it legit GBD?
          08 A2            0275   666                 GBD$B_BID(R2)
             D3      12    0277   667         BNEQ    SCAN                    ; skip it if no good
          20 A2      B7    0279   668         DECW    GBD$W_USECNT(R2)                ; adjust use count
             CE      12    027C   669         BNEQ    SCAN                    ; if <>0 then we could not have
                     027E   670                                                 ;  been writing so don't invalidate
```

```
                10 A2    01    CE  027E    671          MNEGL   #1,GBD$L_VBNSEQNUM(R2)          ; mark invalid by setting sequence n
                         C8    11  0282    672          BRB     SCAN                            ; continue scan of GBPB's
                               0284    673
                               0284    674 ;
                               0284    675 ;  Check to see if last accessor.  If so, then release all buffers in
                               0284    676 ;  in cache.  Disassociate from the global section, and release the GBSB and
                               0284    677 ;  associated lock.
                               0284    678 ;
                               0284    679
00000000'EF     16    0284    680 CLN_UP: JSB     RM$RELEASE_GBL_BUFFERS          ; Flush cache if last accessor.
00000000'EF     16    028A    681         JSB     RM$UNMAP_GBL                    ; Disassociate from section.
                      0290    682 DEQLCK: $DEQ_S  LKID    = GBSB$L_LOCK_ID(R4),-  ; Release the global section lock
                      0290    683                 VALBLK  = GBSB$L_LKSB+8(R4)     ;  writing out the value block.
                   05 029F    684         RSB
                      02A0    685
                      02A0    686         .END
```

```
$$.PSECT_EP                     = 00000000
$$ARGS                          = 0000000C
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
$$T1                            = 00000001
ATR$C_RECATTR                   = 00000004
BDB$B_BID                       = 00000008
BDB$B_BLN                       = 00000009
BDB$B_FLGS                      = 0000000A
BDB$B_REL_VBN                   = 00000048
BDB$B_VAL_VBNS                  = 00000049
BDB$C_BID                       = 0000000C
BDB$C_BLN                       = 00000050
BDB$L_ADDR                      = 00000018
BDB$L_FLINK                     = 00000000
BDB$L_VBN                       = 0000001C
BDB$V_DRT                       = 00000001
BDB$V_VAL                       = 00000000
CHECK_GBL_BUFFERS                 000001AC R      01
CHKDRT                            000000A4 R      01
CLN_UP                            00000284 R      01
CLOSE                             000000F0 R      01
CLOSE_BR                          000000A1 R      01
DEQLCR                            00000290 R      01
DEV$V_RND                       = 0000001C
DOQIO                             00000175 R      01
EXE$GL_SYSID_LOCK                 ******** R    X 01
EXIT                              0000001B R      01
EXTEND_FILE                       00000128 R      01
FIB$B_AGENT_MODE                = 0000002E
FIB$C_LENGTH                    = 00000040
FIB$L_EXSZ                      = 00000018
FIB$L_EXVBN                     = 0000001C
FIB$M_EXTEND                    = 00000080
FIB$W_EXCTL                     = 00000016
FTL$_ENQDEQFAIL                 = FFFFFFF2
FTL$_GETLKIFAIL                 = FFFFFFD5
GBD$B_BID                       = 00000008
GBD$B_BLN                       = 00000009
GBD$C_BID                       = 00000013
GBD$C_BLN                       = 00000028
GBD$L_VBNSEQNUM                 = 00000010
GBD$W_USECNT                    = 00000020
GBH$B_BID                       = 00000008
GBH$B_BLN                       = 00000009
GBH$C_BID                       = 00000011
GBH$C_BLN                       = 00000058
GBPB$B_BID                      = 00000008
GBPB$B_BLN                      = 00000009
GBPB$C_BID                      = 00000015
GBPB$C_BLN                      = 00000028
GBPB$L_FLINK                    = 00000000
GBPB$L_GBD_PTR                  = 00000024
GBPB$W_USERS                    = 0000000C
GBSB$L_LKSB                     = 0000002C
GBSB$L_LOCK_ID                  = 00000030
GBSB$Q_FILENAME                 = 00000000
IFB$B_AGENT_MODE                = 0000004F
IFB$B_BID                       = 00000008
IFB$B_BLN                       = 00000009
IFB$B_ORGCASE                   = 00000023
IFB$B_RFMORG                    = 00000050
IFB$C_BID                       = 0000000B
IFB$C_BLN                       = 00000088
IFB$C_FHAEND                    = 00000066
IFB$L_BDB_FLNK                  = 00000040
IFB$L_EBK                       = 00000074
IFB$L_EBK_DISK                  = 00000058
IFB$L_GBH_PTR                   = 00000088
IFB$L_GBSB_PTR                  = 0000007C
IFB$L_HBK                       = 00000070
IFB$L_HBK_DISK                  = 00000054
IFB$L_IRAB_LNK                  = 0000001C
IFB$L_PRIM_DEV                  = 00000000
IFB$L_SFSB_PTR                  = 00000078
IFB$S_ORG                       = 00000004
IFB$V_ACCESSED                  = 00000025
IFB$V_ORG                       = 00000004
IFB$V_WRTACC                    = 00000030
IFB$W_CHNL                      = 00000020
IMP$C_ASYQIOEFN                 = 0000001F
IMP$C_IOREFN                    = 0000001E
IMP$L_IFABTBL                   = 00000018
IMP$W_ENTPERSEG                 = 00000020
IO$_DEACCESS                    = 00000034
IO$_MODIFY                      = 00000036
IO$_WRITEVBLK                   = 00000030
IRB$B_BID                       = 00000008
IRB$B_BLN                       = 00000009
IRB$C_BID                       = 0000000A
IRB$C_BLN_SEQ                   = 0000006C
IRB$L_CURBDB                    = 00000020
LCK$K_EXMODE                    = 00000005
LCK$M_CANCEL                    = 00000002
LCK$M_SYNCSTS                   = 00000008
LCK$M_SYSTEM                    = 00000010
LCK$M_VALBLK                    = 00000001
LKI$B_STATE_GRMODE              = 00000001
LKI$_STATE                      = 00000101
NXTENT                            0000002D R      01
NXTSEG                            0000002B R      01
NXTSOB                            00000032 R      01
PIO$GW_IIOIMPA                    ******** R    X 01
PIO$GW_PIOIMPA                    ******** R    X 01
PSL$C_EXEC                      = 00000001
QIO$_ASTADR                     = 00000014
QIO$_ASTPRM                     = 00000018
QIO$_CHAN                       = 00000008
QIO$_EFN                        = 00000004
QIO$_FUNC                       = 0000000C
QIO$_IOSB                       = 00000010
QIO$_NARGS                      = 0000000C
```

```
QIO$_P1                         = 0000001C
QIO$_P2                         = 00000020
QIO$_P3                         = 00000024
QIO$_P4                         = 00000028
QIO$_P5                         = 0000002C
QIO$_P6                         = 00000030
RDIFAB                            00000049 R      01
RMS$BUG                           ******** X      01
RMS$LAST_CHANCE                   00000000 RG     01
RMS$RELEASE_GBL_BUFFERS           ******** X      01
RMS$UNMAP_GBL                     ******** X      01
RMS$_NORMAL                     = 00010001
RUNDWN                            00000023 R      01
SCAN                              0000024C R      01
SS$_IVLOCKID                    = 00002124
SYS$CLREF                         ******** GX     01
SYS$DEQ                           ******** GX     01
SYS$ENQW                          ******** GX     01
SYS$GETLKIW                       ******** GX     01
SYS$QIO                           ******** X      01
SYS$WAITFR                        ******** GX     01
WRITE                             000000D5 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
RMS$RMS	000002A0	(672.)	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	29	00:00:00.11	00:00:00.86
Command processing	108	00:00:00.70	00:00:04.50
Pass 1	516	00:00:20.62	00:01:02.55
Symbol table sort	0	00:00:03.15	00:00:06.67
Pass 2	135	00:00:03.78	00:00:13.48
Symbol table output	16	00:00:00.17	00:00:00.30
Psect synopsis output	2	00:00:00.03	00:00:00.26
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	808	00:00:28.56	00:01:28.62

The working set limit was 1800 pages.
112941 bytes (221 pages) of virtual memory were used to buffer the intermediate code.
There were 110 pages of symbol table space allocated to hold 2127 non-local and 10 local symbols.
686 source lines were read in Pass 1, producing 15 object records in Pass 2.
43 pages of virtual memory were used to define 42 macros.

```
                                +----------------------------+
                                ! Macro library statistics .
                                +----------------------------+

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                          12
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                           3
_$255$DUA28:[SYSLIB]STARLET.MLB;2                       23
TOTALS (all libraries)                                  38

2356 GETS were required to define 38 macros.

There were no errors, warnings or information messages.

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