```
RRRRRRRRRRR      MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSS
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
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  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
```

FILEID**RM2PUT

```
RRRRRRRR    MM      MM    222222    PPPPPPPP    UU      UU  TTTTTTTTTT
RRRRRRRR    MM      MM    222222    PPPPPPPP    UU      UU  TTTTTTTTTT
RR      RR  MMMM  MMMM  22      22  PP      PP  UU      UU      TT
RR      RR  MMMM  MMMM  22      22  PP      PP  UU      UU      TT
RR      RR  MM  MM  MM        22    PP      PP  UU      UU      TT
RR      RR  MM  MM  MM        22    PP      PP  UU      UU      TT
RRRRRRRR    MM      MM      22      PPPPPPPP    UU      UU      TT
RRRRRRRR    MM      MM      22      PPPPPPPP    UU      UU      TT
RR  RR      MM      MM    22        PP          UU      UU      TT
RR  RR      MM      MM    22        PP          UU      UU      TT
RR    RR    MM      MM  22          PP          UU      UU      TT          ....
RR    RR    MM      MM  22          PP          UU      UU      TT          ....
RR      RR  MM      MM  2222222222  PP          UUUUUUUUUU      TT          ....
RR      RR  MM      MM  2222222222  PP          UUUUUUUUUU      TT          ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

(2) 145 DECLARATIONS
(3) 178 RM$PUT2 - HIGH LEVEL RELATIVE $PUT
(5) 448 RM$PUTUPD2 - COMMON $PUT AND $UPDATE RELATIVE ROUTINE

```
0000     1          $BEGIN  RM2PUT,000,RM$RMS2,<RELATIVE SPECIFIC PUT>
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 ; Facility: RMS32
0000    29 ;
0000    30 ; Abstract:
0000    31 ;                   This module provides relative file organization
0000    32 ;                   specific processing for the $PUT function.
0000    33 ;
0000    34 ;
0000    35 ; Environment:
0000    36 ;                   Star processor running Starlet exec.
0000    37 ;
0000    38 ; Author: L. F. Laverdure        Creation Date: 7-NOV-1977
0000    39 ;
0000    40 ; Modified By:
0000    41 ;
0000    42 ;       V03-019 JEJ0042         J E Johnson             21-Jun-1984
0000    43 ;                   Correct error in record locking code path that
0000    44 ;                   attempts to deallocate BDB twice if WAT bit is set
0000    45 ;                   and a locking error occurs.
0000    46 ;
0000    47 ;       V03-018 DGB0016         Donald G. Blair         02-Mar-1984
0000    48 ;                   Allocate full-length FIB to support access mode
0000    49 ;                   protected files.
0000    50 ;
0000    51 ;       V03-017 DAS0001         David Solomon           25-Jul-1983
0000    52 ;                   Fix a bug introduced in V03-016 that broke BI journaling
0000    53 ;                   of puts and updates.
0000    54 ;
0000    55 ;       V03-016 KPL0005         Peter Lieberwirth       20-Jun-1983
0000    56 ;                   Change some references to JNLFLG to JNLFLG2.
0000    57 ;
```

```
0000     58 ;       V03-015 KPL0004         Peter Lieberwirth       26-May-1983
0000     59 ;               Fix RJR references for new format.
0000     60 ;
0000     61 ;       V03-014 JWH0206         Jeffrey W. Horn         12-Apr-1983
0000     62 ;               Fix bug in JWH0192 which was causing a call to LOCK
0000     63 ;               instead of QUERY_LOCK when WAT was specified for a $PUT.
0000     64 ;
0000     65 ;       V03-013 KPL0003         Peter Lieberwirth       30-Mar-1983
0000     66 ;               AP is used as a flag to RM$PUTUPD2 to indicate whether the
0000     67 ;               operation is a PUT or an UPDATE. Due to an unrelated change,
0000     68 ;               this flag was sometimes set incorrectly when the operation
0000     69 ;               is a PUT. Fix this by plugging the AP.
0000     70 ;
0000     71 ;       V03-012 RAS0135         Ron Schaefer            17-Mar-1983
0000     72 ;               Corrections to RAS0132 for registers and RJR$_ names.
0000     73 ;
0000     74 ;       V03-011 RAS0132         Ron Schaefer            16-Mar-1983
0000     75 ;               Merge $RMSRDEF into $RJRDEF and revise the interface
0000     76 ;               for RM$WRTJNL for easier use from ISAM.
0000     77 ;
0000     78 ;       V03-010 JWH0192         Jeffrey W. Horn         28-Feb-1983
0000     79 ;               Fix bucheck in $PUT with WAT option.
0000     80 ;
0000     81 ;       V03-009 SPR52290        Jeffrey W. Horn         03-Jan-1983
0000     82 ;               Fix bugcheck in auto-extend with shared files.
0000     83 ;
0000     84 ;       V03-008 KPL0002         Peter Lieberwirth        7-Nov-1982
0000     85 ;               Fix RMSR name again.
0000     86 ;
0000     87 ;       V03-007 JWH0121         Jeffrey W. Horn         04-Nov-1982
0000     88 ;               Fix bug in journal logic that was causing non-journal
0000     89 ;               $PUT to return a zero status.
0000     90 ;
0000     91 ;       V03-006 KPL0001         Peter Lieberwirth       26-Oct-1982
0000     92 ;               Correct size of RJR overhead added to R3 for call to
0000     93 ;               WRTJNL.  Change RMSR names.
0000     94 ;
0000     95 ;       V03-005 JWH0112         Jeffrey W. Horn         06-Oct-1982
0000     96 ;               Implement new RJR format.  Put in code for RU
0000     97 ;               journal support.
0000     98 ;
0000     99 ;       V03-004 KBT0129         Keith B. Thompson       20-Aug-1982
0000    100 ;               Reorganize psects
0000    101 ;
0000    102 ;       V03-003 KBT0117         Keith B. Thompson       6-Aug-1982
0000    103 ;               Remove ref. to set_sifb_adr and correct jeff's revision numbers
0000    104 ;
0000    105 ;       V03-002 There was an un-audited change done by JWH some time
0000    106 ;               around here to fix a bug introduced by JWH0001.
0000    107 ;
0000    108 ;       V03-001 JWH0001         Jeffrey W. Horn         18-May-1982
0000    109 ;               Put in code for BI an AI journal support.
0000    110 ;
0000    111 ;       V02-020 RAS0063         Ron Schaefer            29-Jan-1982
0000    112 ;               Correct probes of the user's key and record buffers.
0000    113 ;
0000    114 ;       V02-019 CDS0077         C. D. Saether           24-Feb-1981     10:35
```

```
0000    115 ;               If really sequential file, specify noread to cache.
0000    116 ;               Note that this works correctly only for 512 byte fixed
0000    117 ;               length records, with the "bucket size" at one block.
0000    118 ;
0000    119 ;       V02-018 CDS0076         C. D. Saether           07-Oct-1980     11:05
0000    120 ;               Release auto-locked record if rm$getrec2_put did not release
0000    121 ;               it (because it was current record).
0000    122 ;
0000    123 ;       V02-017 REFORMAT        K. E. Kinnear           31-Jul-1980     9:05
0000    124 ;
0000    125 ;       V01-016 CDS0075         C. D. Saether           20-Jan-1980     11:40
0000    126 ;               Fix bug so manually locked record is released on error.
0000    127 ;
0000    128 ;       V01-015 CDS0042         C. D. Saether           12-Oct-1979     17:40
0000    129 ;               Update ebk correctly for seq file extend.
0000    130 ;
0000    131 ;       V01-014 JAK0020         J. A. Krycka            11-Sep-1979     10:00
0000    132 ;               Remove network code.
0000    133 ;
0000    134 ;       V01-013 CDS0024         C. D. Saether           27-Jul-79       5:05
0000    135 ;               Fudge up code so it works with shared fix length seq files.
0000    136 ;
0000    137 ;       V01-012 WSK0001         W. S. Koenig            22-Dec-1978     11:20
0000    138 ;               Fixed bug destroying r4 when extend failed.
0000    139 ;
0000    140 ;       V01-011 RAN0003         R. A. Newell            9-Nov-1978      10:56
0000    141 ;               File sharing code enhancements.
0000    142 ;--
0000    143
```

```
                     0000   145          .SBTTL  DECLARATIONS
                     0000   146
                     0000   147 ;
                     0000   148 ; Include Files:
                     0000   149 ;
                     0000   150
                     0000   151 ;
                     0000   152 ; Macros:
                     0000   153 ;
                     0000   154
                     0000   155          $IFBDEF
                     0000   156          $BDBDEF
                     0000   157          $CSHDEF
                     0000   158          $DLCDEF
                     0000   159          $FABDEF
                     0000   160          $FIBDEF
                     0000   161          $RABDEF
                     0000   162          $IRBDEF
                     0000   163          $RMSDEF
                     0000   164          $RJRDEF
                     0000   165          $CJFDEF
                     0000   166
                     0000   167 ;
                     0000   168 ; Equated Symbols:
                     0000   169 ;
                     0000   170
00000020             0000   171          ROP=RAB$L_ROP*8                 ; bit offset to rop field
                     0000   172
                     0000   173 ;
                     0000   174 ; Own Storage:
                     0000   175 ;
                     0000   176
```

```
0000   178          .SBTTL  RM$PUT2 - HIGH LEVEL RELATIVE $PUT
0000   179
0000   180 ;++
0000   181 ; RM$PUT2 -- High Level Relative $PUT.
0000   182 ;
0000   183 ; This module performs the following functions:
0000   184 ;
0000   185 ;       1.  Calls RM$GETREC2_PUT to gain access to the bucket, locking it,
0000   186 ;           and unlocking any record automatically locked.  The unlocking
0000   187 ;           is deferred if key access and the record is the current record
0000   188 ;           to avoid opening a window where the record is unlocked while the
0000   189 ;           bucket is being reaccessed.
0000   190 ;
0000   191 ;       2.  If the return from RM$GETREC2_PUT indicates that the desired
0000   192 ;           record is past the current end of file, calls RM$EXTEND2 to extend
0000   193 ;           the file and tries again
0000   194 ;
0000   195 ;       3.  If manual locking is specified, the record to be written is locked,
0000   196 ;           otherwise the routine merely checks that the record is not locked
0000   197 ;           by another stream. If the record was not unlocked in rm$getrec2_put
0000   198 ;           (unlock_rp still set), it is unlocked at this point.
0000   199 ;
0000   200 ;       4.  The record is checked for non-existence and if so the record is
0000   201 ;           copied to the bucket buffer.
0000   202 ;
0000   203 ;       5.  Access to the bucket is released, causing the buffer to be written
0000   204 ;           unless deferred write has been specified (at open time).
0000   205 ;
0000   206 ; Calling Sequence:
0000   207 ;
0000   208 ;        Entered via case branch from RM$PUT at RM$PUT2.
0000   209 ;
0000   210 ; Input Parameters:
0000   211 ;
0000   212 ;       R11     impure area address
0000   213 ;       R10     ifab addr
0000   214 ;       R9      irab addr
0000   215 ;       R8      rab addr
0000   216 ;
0000   217 ; Implicit Inputs:
0000   218 ;
0000   219 ;       The contents of the rab and related irab and ifab.
0000   220 ;
0000   221 ; Output Parameters:
0000   222 ;
0000   223 ;       R1 thru R7      destroyed
0000   224 ;       R0              status
0000   225 ;
0000   226 ; Implicit Outputs:
0000   227 ;
0000   228 ;       Various fields of the rab are filled in to reflect
0000   229 ;       the status of the operation (see functional spec
0000   230 ;       for details).
0000   231 ;
0000   232 ;       The irab is similarly updated.
0000   233 ;
0000   234 ; Completion Codes:
```

```
0000   235 ;
0000   236 ;       Standard rms (see functional spec).
0000   237 ;
0000   238 ; Side Effects:
0000   239 ;
0000   240 ;       none
0000   241 ;
0000   242 ;--
0000   243
```

```
                     0000   245 RM$PUT2::
                     0000   246         $TSTPT  PUT2
                     0006   247         $CSHFLAGS       LOCK                    ; require lock on bucket
03 6A   38   E1      0009   248         BBC     #IFB$V_SEQFIL,(R10),10$ ; check if really sequential
   53   04   C8      000D   249         BISL2   #CSH$M_NOREAD,R3        ; don't read block if really seq.
     FFED'  30      0010   250 10$:    BSBW    RM$GETREC2_PUT          ; go access bucket
  03 50     E8      0013   251 CHKERR: BLBS    R0,10$                  ; continue if success
    0088    31      0016   252         BRW     CHKEOF                  ; otherwise go check if EOF
                     0019   253 ;
                     0019   254 ;  Handle record locking, if required.
                     0019   255 ;
                     0019   256 ;  If automatic locking (RAB$V_ULK = 0) , need merely check that record
                     0019   257 ;  is not locked since bucket is locked (and no other user could possibly
                     0019   258 ;  lock the record until the bucket is released).
                     0019   259 ;
                     0019   260 ;  If manual unlocking (RAB$V_ULK = 1), must lock the record.
                     0019   261 ;
                     0019   262
53 6A   33   E0      0019   263 10$:    BBS     #IFB$V_NORECLK,(R10),CHKCTL; branch if no locking
51   48 A9   D0      001D   264         MOVL    IRB$L_RP(R9),R1         ; get record #
        52   D4      0021   265         CLRL    R2                      ; zero hi half
                     0023   266 ;
                     0023   267 ;  If record was previously auto-locked and unlocking was deferred to avoid
                     0023   268 ;  and unlocked record window, it is unlocked at this point.
                     0023   269 ;
                     0023   270
07 69   2D   E5      0023   271         BBCC    #IRB$V_UNLOCK_RP,(R9),20$ ; branch if already unlocked
        06   BB      0027   272         PUSHR   #^M<R1,R2>              ; save these
     FFD4'  30      0029   273         BSBW    RM$UNLOCK               ; unlock the record
        06   BA      002C   274         POPR    #^M<R1,R2>              ; restore registers
21 68   32   E0      002E   275 20$:    BBS     #RAB$V_ULK+ROP,(R8),LOCK; branch if manual locking
     FFCB'  30      0032   276         BSBW    RM$QUERY_LCK            ; o.k. to write?
8021 8F   50   B1      0035   277         CMPW    R0,#RMS$_OK_RLK&^XFFFF  ; only read allowed?
          1A   12      003A   278         BNEQ    CHKLCK                  ; branch if not (so far so good)
                     003C   279         RMSERR  RLK                     ; switch status to error
                     0041   280
                     0041   281 ;
                     0041   282 ;  Handle error.
                     0041   283 ;
                     0041   284
54   20 A9   D0      0041   285 CLEAN1: MOVL    IRB$L_CURBDB(R9),R4     ; Update R4 in case BDB was released.
   57   50   D0      0045   286         MOVL    R0,R7                   ; save status code
                     0048   287 CLEAN2:
04 68   32   E1      0048   288         BBC     #RAB$V_ULK+ROP,(R8),10$ ; this record manually locked?
                     004C   289         SSB     #IRB$V_UNLOCK_RP,(R9)   ; yes, make sure it's released
     FFAD'  31      0050   290 10$:    BRW     RM$CLN2_PUT             ; go clean up
                     0053   291
                     0053   292 ;
                     0053   293 ;  Manual locking.  Must lock the record.
                     0053   294 ;
                     0053   295
     FFAA'  30      0053   296 LOCK:   BSBW    RM$LOCK                 ; go lock record
  E8 50     E9      0056   297 CHKLCK: BLBC    R0,CLEAN1               ; branch on failure
8061 8F   50   B1      0059   298         CMPW    R0,#RMS$_OK_WAT&^XFFFF  ; did we wait for it?
          10   12      005E   299         BNEQ    CHKCTL                  ; branch if not
                     0060   300
                     0060   301 ;
```

```
                                0060   302 ; Lock routine stalled, this means that the bucket was
                                0060   303 ; released, go re-access bucket.
                                0060   304 ;
                                0060   305
                                0060   306          $CSHFLAGS       LOCK                    ; require lock on bucket
                03 6A    38     E1     0063   307          BBC     #IFB$V_SEQFIL,(R10),10$ ; check if really sequential
                   53    04     C8     0067   308          BISL2   #CSH$M_NOREAD,R3        ; don't read block if really seq.
                      FF93'     30     006A   309 10$:     BSBW    RM$GETREC2_PUT          ; go re-access bucket
                     D1 50      E9     006D   310          BLBC    R0,CLEAN1               ; get out on error
                                0070   311
                                0070   312 ;
                                0070   313 ;  Locking all set.
                                0070   314 ;  Check for record already existent and if not, copy the record to the buffer.
                                0070   315 ;
                                0070   316
                                0070   317 CHKCTL:
                09 6A    38     E0     0070   318          BBS     #IFB$V_SEQFIL,(R10),10$ ; if seq file no control byte
                   08    65     91     0074   319          CMPB    (R5),#DLC$M_REC         ; does record exist?
                         04     12     0077   320          BNEQ    10$                     ; branch if not (ok to put)
                1D 68    24     E1     0079   321          BBC     #RAB$V_UIF+ROP,(R8),ERRREX; error unless uif bit set
                   5C    01     D0     007D   322 10$:     MOVL    #1,AP                   ; indicate to PUTUPD2 this is a PUT
                      00B2     30     0080   323          BSBW    RM$PUTUPD2              ; go copy record
                     11 57      E9     0083   324          BLBC    R7,30$                  ; branch on error
                                0086   325
                                0086   326          ASSUME  RAB$C_SEQ EQ 0
                                0086   327
                   1E A8    95     0086   328          TSTB    RAB$B_RAC(R8)           ; sequential access?
                         06     12     0089   329          BNEQ    20$                     ; branch if not
         40 A9     01   48 A9     C1     008B   330          ADDL3   IRB$L_RP(R9),#1,IRB$L_NRP(R9); yes - set nrp from rp + 1
                   48 A9    D4     0091   331 20$:     CLRL    IRB$L_RP(R9)            ; show no current record
                    FF69'     31     0094   332          BRW     RM$RL52                 ; go finish up
                    FF66'     31     0097   333 30$:     BRW     RM$CLN2_PUT             ; clean up on error
                                009A   334
                                009A   335 ;
                                009A   336 ;  Record already exists.  Declare error and go clean up.
                                009A   337 ;
                                009A   338
                                009A   339 ERRREX:
                                009A   340          RMSERR  REX,R7                   ; set error code
                         A7     11     009F   341          BRB     CLEAN2                  ; go clean up
                                00A1   342
                                00A1   343 ;
                                00A1   344 ;  Check if error from RM$GETREC2_PUT is due to eof, and if so extend the file.
                                00A1   345 ;
                                00A1   346
                                00A1   347 CHKEOF:
      827A 8F    50     B1     00A1   348          CMPW    R0,#RMS$_EOF&^XFFFF     ; is error = eof?
                   7A     12     00A6   349          BNEQ    10$                     ; branch if not
                56 52     D0     00A8   350          MOVL    R2,R6                   ; save desired hi vbn + 1
                      FF52'     30     00AB   351          BSBW    RM$LOCK_PROLOG          ; lock vbn 1
                     71 50      E9     00AE   352          BLBC    R0,10$                  ; branch on error
                  6C AA    54     D0     00B1   353          MOVL    R4,IFB$L_LOCK_BDB(R10)  ; save bdb addr
                                00B5   354
                                00B5   355 ;
                                00B5   356 ;  Prolog is now interlocked, thus preventing other extends.
                                00B5   357 ;  Check that extend is still required.
                                00B5   358 ;
```

```
                                00B5   359
                56   74 AA      D1   00B5   360          CMPL    IFB$L_EBK(R10),R6       ; still need to extend eof?
                                6A   1E   00B9   361          BGEQU   20$             ; branch if not
                                56   D7   00BB   362          DECL    R6              ; adjust for hbk
                56   70 AA      C2   00BD   363          SUBL2   IFB$L_HBK(R10),R6       ; compute # of blocks needed
                                67   1B   00C1   364          BLEQU   25$             ; branch if none (need only format)
                                     00C3   365
                                     00C3   366 ;
                                     00C3   367 ;  Allocate a fib to do the extend.
                                     00C3   368 ;
                                     00C3   369
                                54   DD   00C3   370          PUSHL   R4              ; save lock bdb addr around calls
                52   40 8F      9A   00C5   371          MOVZBL  #FIB$C_LENGTH,R2        ; size of fib
                                FF34'   30   00C9   372          BSBW    RM$GETSPC1      ; go allocate fib
                                4C   50   E9   00CC   373          BLBC    R0,8$           ; branch on failure
          18 A1   4C   AA      3C   00CF   374          MOVZWL  IFB$W_RTDEQ(R10),FIB$L_EXSZ(R1); set default extend size
                                04   12   00D4   375          BNEQ    2$              ; branch if non-zero
             16 A1   08   88   00D6   376          BISB2   #FIB$M_ALDEF,FIB$W_EXCTL(R1); flag maximize with vol. default
                56   18 A1      D1   00DA   377 2$:      CMPL    FIB$L_EXSZ(R1),R6       ; is default > # blocks needed?
                                04   1E   00DE   378          BGEQU   4$              ; branch if yes
                18 A1   56      D0   00E0   379          MOVL    R6,FIB$L_EXSZ(R1)       ; no - use required extend size
                                     00E4   380 4$:
                                     00E4   381
                                     00E4   382 ;
                                     00E4   383 ;  Do the extend.
                                     00E4   384 ;
                                     00E4   385
                20 A9      D4   00E4   386          CLRL    IRB$L_CURBDB(R9)        ; zero current bdb
                                FF16'   30   00E7   387          BSBW    RM$EXTEND0_ALT  ; do extend and deallocate fib
                                2E   50   E9   00EA   388          BLBC    R0,8$           ; branch on error
                                     00ED   389
                                     00ED   390 ;
                                     00ED   391 ; If extend worked we can get rid of bdb addr (r4) on top of stack.
                                     00ED   392 ;
                                     00ED   393
                                8E   D5   00ED   394          TSTL    (SP)+
                                     00EF   395
                                     00EF   396 ;
                                     00EF   397 ;  Format the buckets (i.e., write zeroes) and update prolog.
                                     00EF   398 ;
                                     00EF   399
                                     00EF   400 15$:
             16 6A      38      E0   00EF   401          BBS     #IFB$V_SEQFIL,(R10),7$  ; if seq don't zero
                                FF0A'   30   00F3   402          BSBW    RM$FMT_BKT2     ; write zeroed blocks
                                FF07'   30   00F6   403          BSBW    RM$UPD_PROLOG2  ; update prolog
                                26   50   E9   00F9   404          BLBC    R0,10$          ; branch on error
                                     00FC   405 3$:       $CSHFLAGS       LOCK            ; specify lock required
                51   44 A9      D0   00FF   406 5$:       MOVL    IRB$L_CURVBN(R9),R1     ; set vbn
                                FEFA'   30   0103   407          BSBW    RM$READBKT2     ; go access bucket
                                FF0A   31   0106   408          BRW     CHKERR          ; and try again
                                     0109   409 7$:
                54   6C AA      D0   0109   410          MOVL    IFB$L_LOCK_BDB(R10),R4  ; restore r4
                                FEF0'   30   010D   411          BSBW    RM$SETHEBK      ; update eof
          74 AA   01   44 A9      C1   0110   412          ADDL3   IRB$L_CURVBN(R9),#1,IFB$L_EBK(R10); just make ebk beyond block
                                     0116   413                                          ; asked for if seq file.
                                     0116   414
                                     0116   415          $CSHFLAGS <LOCK,NOREAD>          ; no need to read block if seq file.
```

```
                E4   11  0119   416          BRB     5$                      ; try again
                         011B   417
                         011B   418 ;
                         011B   419 ;  Error has occurred allocating a fib or extending the file.
                         011B   420 ;
                         011B   421
                         011B   422 8$:
                10   BA  011B   423          POPR    #^M<R4>                 ; restore bdb addr for cleanup
              FEE0'  30  011D   424          BSBW    RM$RLSPLG               ; release lock on prolog
                54   D4  0120   425          CLRL    R4                      ; don't release bdb twice
              FF1C   31  0122   426 10$:     BRW     CLEAN1                  ; clean up on error
                         0125   427
                         0125   428 ;
                         0125   429 ;  Another user has done an extend since we last checked the eof data
                         0125   430 ;  (hard to believe he could sneak in that window, isn't it?)
                         0125   431 ;
                         0125   432 ;  Therefore, our work is already done.  We need merely unlock the prolog
                         0125   433 ;  and go try our put again.
                         0125   434 ;
                         0125   435
              FED8'  30  0125   436 20$:     BSBW    RM$RLSPLG               ; unlock the prolog
                D2   11  0128   437          BRB     3$                      ; continue with put
                         012A   438
                         012A   439 ;
                         012A   440 ;  File needs no extending, only formatting.
                         012A   441 ;
                         012A   442
56   70 AA      01   C1  012A   443 25$:     ADDL3   #1,IFB$L_HBK(R10),R6    ; set end vbn of extent + 1
     51    74 AA   D0  012F   444          MOVL    IFB$L_EBK(R10),R1       ; and start vbn of extent
                BA   11  0133   445          BRB     15$                     ; go format buckets
                         0135   446
```

```
0135    448          .SBTTL  RM$PUTUPD2 - COMMON $PUT AND $UPDATE RELATIVE ROUTINE
0135    449
0135    450 ;++
0135    451 ; RM$PUTUPD2 -- Common $PUT and $UPDATE Relative Routine.
0135    452 ;
0135    453 ;  This routine:
0135    454 ;
0135    455 ;       1. Saves r0 status code in r7.
0135    456 ;
0135    457 ;       2. Verifies the rsz and rbf parameters, as well as rhb if rfm=vfc.
0135    458 ;
0135    459 ;       3. Set the delete control byte to say record exists.
0135    460 ;
0135    461 ;       4. Store record size unless rfm=fix.
0135    462 ;
0135    463 ;       5. If rfm=vfc, copy the rhb to buffer.
0135    464 ;
0135    465 ;       6. Copy the record to the buffer and set the valid and dirty
0135    466 ;          buffer flags.
0135    467 ;
0135    468 ;  Calling Sequence:
0135    469 ;
0135    470 ;       BSBW    RM$PUTUPD2
0135    471 ;
0135    472 ;  Input Parameters:
0135    473 ;
0135    474 ;       AP      non 0 if called from $put, 0 if from $update
0135    475 ;       R8-R11  same as for entry at rm$put2
0135    476 ;       R5      address of record in bucket buffer
0135    477 ;       R4      bdb address
0135    478 ;       R0      status code
0135    479 ;
0135    480 ;  Implicit Inputs:
0135    481 ;
0135    482 ;    The contents of the various control blocks, in particular:
0135    483 ;
0135    484 ;       RAB$W_RSZ       record size
0135    485 ;       RAB$L_RBF       record address
0135    486 ;       RAB$L_RHB       record header buffer address if rfm=vfc
0135    487 ;       IFB$B_RFM       record format
0135    488 ;       IFB$W_MRS       maximum record length
0135    489 ;       IFB$B_FSZ       fixed header size if rfm=vfc
0135    490 ;
0135    491 ;  Output Parameters:
0135    492 ;
0135    493 ;       R7              status code
0135    494 ;       R0-R3,R5,R6     destroyed
0135    495 ;
0135    496 ;  Implicit Outputs:
0135    497 ;
0135    498 ;       none.
0135    499 ;
0135    500 ;  Completion Codes:
0135    501 ;
0135    502 ;       Standard rms, in particular the code from r0 on input or rsz, rbf,
0135    503 ;       or rhb.
0135    504 ;
```

```
0135    505 ;  Side Effects:
0135    506 ;
0135    507 ;       none.
0135    508 ;--
0135    509
```

```
                        0135   511 RM$PUTUPD2::
          57    50      D0 0135   512          MOVL    R0,R7                                   ; copy status code
06 00A0   CA    02      E0 0138   513          BBS     #IFB$V_BI,IFB$B_JNLFLG(R10),5$          ; BI journaling on?
42 00A2   CA    02      E1 013E   514          BBC     #IFB$V_RUP,IFB$B_JNLFLG2(R10),40$       ; RUP? If no BI/RUP, skip this
                30      BB 0144   515 5$:      PUSHR   #^M<R4,R5>                              ; save R4,R5
                   0124 30 0146   516          BSBW    MAKEJNL                                 ; set up journal record
                5C      D5 0149   517          TSTL    AP                                      ; doing $PUT?
                06      13 014B   518          BEQL    10$                                     ; branch if update
14 A6      0048 8F      B0 014D   519          MOVW    #RJR$C_RECLEN,BDB$W_NUMB(R6)            ; don't write empty cell
                54      6E D0 0153   520 10$:    MOVL    (SP),R4                                 ; restore R4
11 00A0   CA    02      E1 0156   521          BBC     #IFB$V_BI,IFB$B_JNLFLG(R10),20$         ; branch if no BI
                56      DD 015C   522          PUSHL   R6                                      ; jBDB arg
                7E      02 9A 015E   523          MOVZBL  #CJF$_BI,-(SP)                          ; specify BI
             00000000'EF    16 0161   524          JSB     RM$WRTJNL                               ; write journal record
                5E      08 C0 0167   525          ADDL2   #8,SP                                   ; discard arglist
                  14    50 E9 016A   526          BLBC    R0,30$                                  ; get out on error
0E 00A2   CA    02      E1 016D   527 20$:     BBC     #IFB$V_RUP,IFB$B_JNLFLG2(R10),30$; branch if not RUP
                56      DD 0173   528          PUSHL   R6                                      ; jBDB arg
                7E      01 9A 0175   529          MOVZBL  #CJF$_RU,-(SP)                          ; specify BI
             00000000'EF    16 0178   530          JSB     RM$WRTJNL                               ; write journal record
                5E      08 C0 017E   531          ADDL2   #8,SP                                   ; discard arglist
                30      BA 0181   532 30$:     POPR    #^M<R4,R5>
                   6D   50 E9 0183   533          BLBC    R0,ERRJNL                               ; get out on error
                        0186   534
          56    22 A8   3C 0186   535 40$:     MOVZWL  RAB$W_RSZ(R8),R6                        ; get record size
          01    50 AA   91 018A   536          CMPB    IFB$B_RFMORG(R10),#FAB$C_FIX            ; rfm = fix?
                06      13 018E   537          BEQL    50$                                     ; branch of yes
          56    60 AA   B1 0190   538          CMPW    IFB$W_MRS(R10),R6                       ; record too long?
                06      1E 0194   539          BGEQU   60$                                     ; branch if ok
                        0196   540                                                          ;  (else fall thru - will be checked
          56    60 AA   B1 0196   541 50$:     CMPW    IFB$W_MRS(R10),R6                       ; rsz = fixed record length?
                4B      12 019A   542          BNEQ    ERRRSZ                                  ; branch if not
          03 6A   38      E1 019C   543 60$:     BBC     #IFB$V_SEQFIL,(R10),CHKVFC
                   0082 31 01A0   544          BRW     SAVR45                                  ; let's move record and be done
          03    50 AA   91 01A3   545 CHKVFC:  CMPB    IFB$B_RFMORG(R10),#FAB$C_VFC            ; rfm = vfc?
                70      12 01A7   546          BNEQ    SETCTL                                  ; branch if not
                        01A9   547
                        01A9   548 ;
                        01A9   549 ;  Record format is vfc.
                        01A9   550 ;
                        01A9   551 ;  Probe the record header buffer and copy to bucket.
                        01A9   552 ;
                        01A9   553
          51    5F AA   9A 01A9   554          MOVZBL  IFB$B_FSZ(R10),R1               ; get fixed header size
          50    2C A8   D0 01AD   555          MOVL    RAB$L_RHB(R8),R0                ; get the rhb address
                07      13 01B1   556          BEQL    10$                             ; branch if none
                        01B3   557          IFNORD  R1,(R0),ERRRHB,IRB$B_MODE(R9); branch if not readable
             85    08   90 01BA   558 10$:     MOVB    #DLC$M_REC,(R5)+                ; say record exists
85           51    56   A1 01BD   559          ADDW3   R6,R1,(R5)+                     ; set rec length = fixed + var
                30      BB 01C1   560          PUSHR   #^M<R4,R5>                      ; save R4, R5
                50      D5 01C3   561          TSTL    R0                              ; rhb speced?
                06      13 01C5   562          BEQL    20$                             ; branch if not
65          60    51    28 01C7   563          MOVC3   R1,(R0),(R5)                    ; copy rhb
                0F      11 01CB   564          BRB     40$
                        01CD   565
                        01CD   566 ;
                        01CD   567 ;  Rhb = 0.  Zero the header if doing $PUT, skip it if $UPDATE.
```

```
                                01CD    568 ;
                                01CD    569
                          5C    D5  01CD    570 20$:    TSTL    AP                          ; doing $put?
                          05    12  01CF    571         BNEQ    30$                         ; branch if yes
                    55    51    C0  01D1    572         ADDL2   R1,R5                       ; skip over header
                          09    11  01D4    573         BRB     50$
          65  51  00    6E    00  2C  01D6    574 30$:    MOVC5   #0,(SP),#0,R1,(R5)          ; zero the header
                    55    53    D0  01DC    575 40$:    MOVL    R3,R5                       ; update buffer address
                          46    11  01DF    576 50$:    BRB     MOVREC                      ; go move record
                                01E1    577
                                01E1    578
                                01E1    579 ;
                                01E1    580 ;  Handle errors.
                                01E1    581 ;
                                01E1    582
                                01E1    583 ERRRHB:
                                01E1    584         RMSERR  RHB,R7                      ; bad record header buffer
                          05    01E6    585         RSB
                                01E7    586
                                01E7    587 ERRRSZ:
                                01E7    588         RMSERR  RSZ,R7                      ; invalid record length
                          05    01EC    589         RSB
                                01ED    590
                                01ED    591 ERRRBF:
                                01ED    592         RMSERR  RBF,R7                      ; invalid record header buffer
                          05    01F2    593         RSB
                                01F3    594
                                01F3    595 ERRJNL:
                    57    50    D0  01F3    596         MOVL    R0,R7
                          05    01F6    597         RSB
                                01F7    598
                                01F7    599 ;
                                01F7    600 ;  Probe readability of all pages ( > 2) of user record.
                                01F7    601 ;
                                01F7    602
                                01F7    603 LONG_PROBE:
                    50    56    D0  01F7    604         MOVL    R6,R0                       ; copy buffer length
                    51    53    D0  01FA    605         MOVL    R3,R1                       ; and address
          52    FE00  8F    32  01FD    606         CVTWL   #-512,R2                    ; set address constant
                                0202    607 10$:    IFNORD  R0,(R1),ERRRBF,IRB$B_MODE(R9); branch if not readable
                    51    52    C2  0209    608         SUBL2   R2,R1                       ; get address next page
               50   6042  3E    020C    609         MOVAW   (R0)[R2],R0                 ; adjust remaining length
                          F0    14  0210    610         BGTR    10$                         ; loop if more to do
                    50    52    C2  0212    611         SUBL2   R2,R0                       ; need to handle last page?
                          EB    14  0215    612         BGTR    10$                         ; branch if yes
                          24    11  0217    613         BRB     MOVREC1                     ; rejoin main sequence
                                0219    614
```

```
                                0219   616
                                0219   617 ;
                                0219   618 ;  Set "record exists" into control byte, store the record size if var record
                                0219   619 ;  format, and move the record.
                                0219   620 ;
                                0219   621
        85    08    90        0219   622 SETCTL: MOVB    #DLC$M_REC,(R5)+                ; say record exists
     02   50 AA    91        021C   623          CMPB    IFB$B_RFMORG(R10),#FAB$C_VAR; variable len rfm?
                    03  12    0220   624          BNEQ    SAVR45                          ; branch if not
        85    56    B0        0222   625          MOVW    R6,(R5)+                        ; store record length
                    30  BB    0225   626 SAVR45: PUSHR   #^M<R4,R5>                     ; save R4, R5
     53   28 B8              DE  0227   627 MOVREC: MOVAL   @RAB$L_RBF(R8),R3           ; get buffer addr
                    56  B5    022B   628          TSTW    R6                              ; rsz = 0?
                    0E  13    022D   629          BEQL    MOVREC1                         ; branch if yes
   0200 8F          56  B1    022F   630          CMPW    R6,#512                         ; record > 2 pages in length?
                    C1  1A    0234   631          BGTRU   LONG_PROBE                      ; branch if yes
                              0236   632          IFNORD  R6,(R3),ERRRBF,IRB$B_MODE(R9); branch if not readable
                              023D   633 MOVREC1:
        65    63    56        28  023D   634          MOVC3   R6,(R3),(R5)                    ; move the record
              54    6E        D0  0241   635          MOVL    (SP),R4                         ; restore R4
           0A A4    03        88  0244   636          BISB    #BDB$M_VAL!BDB$M_DRT,BDB$B_FLGS(R4); say valid and dirty
     1C 00A0 CA     03        E1  0248   637          BBC     #IFB$V_AI,IFB$B_JNLFLG(R10),10$ ; branch if not AI journaling
55   18 A4    4C    A9        C1  024E   638          ADDL3   IRB$L_RP_OFF(R9),BDB$L_ADDR(R4),R5 ; get cell address
                    17  10    0254   639          BSBB    MAKEJNL                         ; set up journal record
              54    6E        D0  0256   640          MOVL    (SP),R4                         ; restore R4
                    56  DD    0259   641          PUSHL   R6                              ; jBDB arg
              7E    03        9A  025B   642          MOVZBL  #CJF$_AI,-(SP)                  ; specify BI
     00000000'EF    16        025E   643          JSB     RM$WRTJNL                       ; write journal record
              5E    08        C0  0264   644          ADDL2   #8,SP                           ; discard arglist
              57    50        D0  0267   645          MOVL    R0,R7                           ; set status code
                    30  BA    026A   646 10$:     POPR    #^M<R4,R5>                      ; restore R4, R5
                    05  026C   647 20$:     RSB
```

```
                     026D   649 ;++
                     026D   650 ;       Subroutine to construct journal entry
                     026D   651 ;
                     026D   652 ;
                     026D   653 ;       Input:
                     026D   654 ;               R5      -       Cell to journal
                     026D   655 ;
                     026D   656 ;       Output:
                     026D   657 ;               R6      -       Addr of journaling BDB
                     026D   658 ;               Destroys        R1,R2,R4-R5
                     026D   659 ;--
                     026D   660
                     026D   661
                     026D   662 MAKEJNL:
    56   30 A9   D0  026D   663         MOVL    IRB$L_JNLBDB(R9),R6                     ; get journling BDB
    52   18 A6   D0  0271   664         MOVL    BDB$L_ADDR(R6),R2                       ; get journling buffer
40 A2   48 A9   D0  0275   665         MOVL    IRB$L_RP(R9),RJR$L_RRN(R2)              ; fill in relative record num
    03 A2   02   90  027A   666         MOVB    #RJR$C_RECORD,RJR$B_ENTRY_TYPE(R2) ; RJR type
                     027E   667
                     027E   668         ASSUME  RJR$B_OPER      EQ      RJR$B_ORG+1
                     027E   669
      1301 8F   B0  027E   670         MOVW    #<RJR$_PUT@8 + RJR$C_REL>,-
        04 A2       0282   671                 RJR$B_ORG(R2)                           ; fill in file type & oper
           5C   D5  0284   672         TSTL    AP                                      ; doing $PUT?
           04   12  0286   673         BNEQ    10$                                     ; branch if so
    05 A2   1C   90  0288   674         MOVB    #RJR$_UPDATE,RJR$B_OPER(R2)             ; indicate $UPDATE
46 A2   62 A9   B0  028C   675 10$:     MOVW    IRB$W_CSIZ(R9),RJR$W_RSIZE(R2)  ; set cell size
0048 8F   62 A9   A1  0291   676         ADDW3   IRB$W_CSIZ(R9),#RJR$C_RECLEN,-
           14 A6       0297   677                 BDB$W_NUMB(R6)                          ; set journal record size
    65   62 A9   28  0299   678         MOVC3   IRB$W_CSIZ(R9),(R5),-
           48 A2       029D   679                 RJR$T_RIMAGE(R2)                        ; copy entire cell
               05   029F   680         RSB
                    02A0   681
                    02A0   682         .END
```

```
$$.PSECT_EP                     = 00000000
$$.TMP                          = 00000005
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
BDB$B_FLGS                      = 0000000A
BDB$L_ADDR                      = 00000018
BDB$M_DRT                       = 00000002
BDB$M_VAL                       = 00000001
BDB$W_NUMB                      = 00000014
CHKCTL                            00000070  R      01
CHKEOF                            000000A1  R      01
CHKERR                            00000013  R      01
CHKLCK                            00000056  R      01
CHKVFC                            000001A3  R      01
CJF$_AI                         = 00000003
CJF$_BI                         = 00000002
CJF$_RU                         = 00000001
CLEAN1                            00000041  R      01
CLEAN2                            00000048  R      01
CSH$M_LOCK                      = 00000001
CSH$M_NOBUFFER                  = 00000008
CSH$M_NOREAD                    = 00000004
DLC$M_REC                       = 00000008
ERRJNL                            000001F3  R      01
ERRRBF                            000001ED  R      01
ERRREX                            0000009A  R      01
ERRRHB                            000001E1  R      01
ERRRSZ                            000001E7  R      01
FAB$C_FIX                       = 00000001
FAB$C_VAR                       = 00000002
FAB$C_VFC                       = 00000003
FIB$C_LENGTH                    = 00000040
FIB$L_EXSZ                      = 00000018
FIB$M_ALDEF                     = 00000008
FIB$W_EXCTL                     = 00000016
IFB$B_FSZ                       = 0000005F
IFB$B_JNLFLG                    = 000000A0
IFB$B_JNLFLG2                   = 000000A2
IFB$B_RFMORG                    = 00000050
IFB$L_EBK                       = 00000074
IFB$L_HBK                       = 00000070
IFB$L_LOCK_BDB                  = 0000006C
IFB$V_AI                        = 00000003
IFB$V_BI                        = 00000002
IFB$V_NORECLK                   = 00000033
IFB$V_RUP                       = 00000002
IFB$V_SEQFIL                    = 00000038
IFB$W_MRS                       = 00000060
IFB$W_RTDEQ                     = 0000004C
IRB$B_MODE                      = 0000000A
IRB$L_CURBDB                    = 00000020
IRB$L_CURVBN                    = 00000044
IRB$L_JNLBDB                    = 00000030
IRB$L_NRP                       = 00000040
IRB$L_RP                        = 00000048
IRB$L_RP_OFF                    = 0000004C
IRB$V_UNLOCK_RP                 = 0000002D
IRB$W_CSIZ                      = 00000062
LOCK                              00000053  R      01
LONG_PROBE                        000001F7  R      01
MAKEJNL                           0000026D  R      01
MOVREC                            00000227  R      01
MOVREC1                           0000023D  R      01
PIO$A_TRACE                       ********    X    01
RAB$B_RAC                       = 0000001E
RAB$C_SEQ                       = 00000000
RAB$L_RBF                       = 00000028
RAB$L_RHB                       = 0000002C
RAB$L_ROP                       = 00000004
RAB$V_UIF                       = 00000004
RAB$V_ULK                       = 00000012
RAB$W_RSZ                       = 00000022
RJR$B_ENTRY_TYPE                = 00000003
RJR$B_OPER                      = 00000005
RJR$B_ORG                       = 00000004
RJR$C_RECLEN                    = 00000048
RJR$C_RECORD                    = 00000002
RJR$C_REL                       = 00000001
RJR$L_RRN                       = 00000040
RJR$T_RIMAGE                    = 00000048
RJR$W_RSIZE                     = 00000046
RJR$_PUT                        = 00000013
RJR$_UPDATE                     = 0000001C
RM$CLN2_PUT                       ********    X    01
RM$EXTEND0_ALT                    ********    X    01
RM$FMT_BKT2                       ********    X    01
RM$GETREC2_PUT                    ********    X    01
RM$GETSPC1                        ********    X    01
RM$LOCK                           ********    X    01
RM$LOCK_PROLOG                    ********    X    01
RM$PUT2                           00000000  RG     01
RM$PUTUPD2                        00000135  RG     01
RM$QUERY_LCK                      ********    X    01
RM$READBKT2                       ********    X    01
RM$RLS2                           ********    X    01
RM$RLSPLG                         ********    X    01
RM$SETHEBK                        ********    X    01
RM$UNLOCK                         ********    X    01
RM$UPD_PROLOG2                    ********    X    01
RM$WRTJNL                         ********    X    01
RMS$_EOF                        = 0001827A
RMS$_OK_RLK                     = 00018021
RMS$_OK_WAT                     = 00018061
RMS$_RBF                        = 00018654
RMS$_REX                        = 000182A2
RMS$_RHB                        = 0001866C
RMS$_RLK                        = 000182AA
RMS$_RSZ                        = 000186A4
ROP                             = 00000020
SAVR45                            00000225  R      01
SETCTL                            00000219  R      01
TPT$L_PUT2                        ********    X    01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RM$RMS2	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	38	00:00:00.08	00:00:00.65
Command processing	112	00:00:00.71	00:00:04.29
Pass 1	385	00:00:14.11	00:00:44.48
Symbol table sort	0	00:00:02.08	00:00:03.97
Pass 2	131	00:00:02.85	00:00:09.51
Symbol table output	13	00:00:00.10	00:00:00.13
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	683	00:00:19.96	00:01:03.18

The working set limit was 1650 pages.
79795 bytes (156 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1482 non-local and 35 local symbols.
682 source lines were read in Pass 1, producing 15 object records in Pass 2.
27 pages of virtual memory were used to define 26 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	15
_255DUA28:[SYS.OBJ]LIB.MLB;1	1
_255DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	22

1603 GETS were required to define 22 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:RM2PUT/OBJ=OBJ$:RM2PUT MSRC$:RM2PUT/UPDATE=(ENH$:RM2PUT)+EXECML$/LIB+LIB$:RMS/LIB
```

RM2UPDDEL
LIS

RM3CLOSE
LIS

RM3ALLBKT
LIS

RM2CREATE
LIS

RM2GET
LIS

RM2PUT
LIS

RM3BKTIO
LIS

RM3BKTSPL
LIS

RM3CMPKEY
LIS

RM3CMPRSS
LIS

RM2EXTEND
LIS

RM2FMTBKT
LIS

RM3BUG
LIS

RM2OPEN
LIS