```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

```
**FILE**ID**RM3RRV
```

```
RRRRRRRR    MM      MM    333333    RRRRRRRR    RRRRRRRR    VV      VV
RRRRRRRR    MM      MM    333333    RRRRRRRR    RRRRRRRR    VV      VV
RR      RR  MMMM  MMMM  33      33  RR      RR  RR      RR  VV      VV
RR      RR  MMMM  MMMM  33      33  RR      RR  RR      RR  VV      VV
RR      RR  MM  MM  MM          33  RR      RR  RR      RR  VV      VV
RR      RR  MM  MM  MM          33  RR      RR  RR      RR  VV      VV
RRRRRRRR    MM      MM        33    RRRRRRRR    RRRRRRRR    VV      VV
RRRRRRRR    MM      MM        33    RRRRRRRR    RRRRRRRR    VV      VV
RR  RR      MM      MM          33  RR  RR      RR  RR      VV      VV
RR  RR      MM      MM          33  RR  RR      RR  RR      VV      VV
RR    RR    MM      MM  33      33  RR    RR    RR    RR      VV  VV        ....
RR    RR    MM      MM  33      33  RR    RR    RR    RR      VV  VV        ....
RR      RR  MM      MM    333333    RR      RR  RR      RR      VV          ....
RR      RR  MM      MM    333333    RR      RR  RR      RR      VV          ....

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
```

```
0 MODULE RM3RRV (LANGUAGE (BLISS32) ,
0                IDENT = 'V04-000'
0                ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1
1 !++
1 !
1 ! FACILITY:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
1 !
1 ! ABSTRACT:
1 !                ROUTINES TO UPDATE RRV'S
1 !
1 !
1 ! ENVIRONMENT:
1 !
1 !               VAX/VMS OPERATING SYSTEM
1 !
1 !--
1
1 !
1 ! AUTHOR:        Wendy Koenig        CREATION DATE:             25-JUL-78  15:24
1 !
1 ! Modified by:
1 !
1 !       V03-012 JWT0149         Jim Teague              19-Jan-1984
1 !               Correct JWT0146.  Actually, in the event that the new
1 !               record (for a $PUT) is to be inserted before a deleted
1 !               record, NXTID should be incremented.  Falling through
1 !               the logic is correct as long as REC_ADDR is positioned
1 !               to the next record (just after the deleted record).
1 !               What was incorrect before was the case where the new
1 !               record caused a 3-bkt split, and the new record ended
1 !               up in a bucket of its own (middle bkt).  As rrvs were
```

```
1 !                     created for the new right bucket, the "if .nxtid nequ 1"
1 !                     test passed BECAUSE THE NEW RIGHT BUCKET WAS A RECLAIMED
1 !                     BUCKET!  Thus, nxtid got incremented once too much.
1 !                     The fix is to remove the "if .nxtid nequ 1" test, because
1 !                     the rest of the test is quite sufficient to insure correct
1 !                     id assignment.
1 !
1 !             V03-011 JWT0146         Jim Teague              05-Dec-1983
1 !                     Fix an RRV misdirection problem for the case of a
1 !                     record $PUT before a deleted record.  The record id
1 !                     of a displaced record was incremented once too much,
1 !                     because when the record being inserted will end up
1 !                     in the new bucket, an id is skipped for it when
1 !                     building RRVs to point to the new bucket.  That's all
1 !                     cool, but when pos_ins eql rec_addr (the position for
1 !                     insert is the current record), and the current record
1 !                     is a deleted record, RMS increments the record id (NXTID)
1 !                     and then falls almost immediately through to the bottom
1 !                     of the WHILE loop, where it will increment the new-bucket
1 !                     record id again.
1 !
1 !             V03-010 MCN0014         Maria del C. Nasr       22-Mar-1983
1 !                     More changes in the linkages
1 !
1 !             V03-009 MCN0013         Maria del C. Nasr       28-Feb-1983
1 !                     Reorganize linkages
1 !
1 !             V03-008 TMK0005         Todd M. Katz            27-Jan-1983
1 !                     Add support for RMS Journalling and RU ROLLBACK Recovery of
1 !                     ISAM files. This involves adding a flag byte (with one bit
1 !                     defined - TBL$V_RU_DELETE) to each prologue 3 RRV table entry,
1 !                     setting the bit within RM$UPDATE_RRV for each entry that refers
1 !                     to a RU_DELETEd primary data record whose RRV is to be updated,
1 !                     and referencing the bit within RM$UPDATE_RRV2 before deciding
1 !                     whether to return an RVU error or not. If RMS is unable to
1 !                     position to a RRV and the bit is clear, RMS returns a RVU error
1 !                     as before. However, if RMS is unable to position to a RRV and
1 !                     the bit is set, then RMS assumes that the Recovery Unit in
1 !                     which the RRV was deleted has successfully completed, that the
1 !                     space occupied by the RRV was reclaimed as part of a general
1 !                     space reclamation of the bucket, and that there is no need to
1 !                     return an RVU error in this case.
1 !
1 !             V03-007 TMK0004         Todd M. Katz            26-Jan-1983
1 !                     Fix two bugs in RM$UPDATE_RRV.
1 !
1 !                     At one point in this routine a reference was made to a bit in
1 !                     the current record even though RMS may currently be positioned
1 !                     to the end of the bucket and there is no current record to
1 !                     reference. The fix is to make sure that the current record
1 !                     position is not at the end of the bucket before referencing
1 !                     this bit.
1 !
1 !                     The second bug is seen in prologue 3 files during $UPDATEs
1 !                     when the record being updated is in its original bucket and is
1 !                     to move into a new bucket as the result of the split, and the
1 !                     record which follows this record in the bucket splitting is
```

```
                        marked deleted. In this case RMS is not creating a RRV for the
                        record being modified in the old bucket. To fix this, RMS must
                        make sure that if it currently is at the position of insertion
                        of the updated record in its bucket scan, that an RRV is
                        created for this record in the orginal bucket, if the updated
                        record was in its original bucket to begin with.

                V03-006 TMK0003         Todd M. Katz            10-Jan-1983
                        In RM$UPDATE_RRV2, always release the scratch buffer that was
                        used to hold the table of RRVs to be updated. The BDB for this
                        scratch buffer is to be found in IRB$L_NXTBDB. Formerly this
                        buffer was bot being released if the data bucket split occurred
                        because of an $UPDATE and there are old SIDRs to delete;
                        however, a re-writing of $UPDATE has changed this requirement.

                V03-005 KBT0233         Keith B. Thompson       23-Aug-1982
                        Reorganize psects

                V03-004 TMK0002         Todd M. Katz            06-Aug-1982
                        The RMS cluster solution for next record positioning mandates
                        that when duplicates are allowed, and a record is deleted,
                        the space occupied by that record can not be completely
                        recovered either during the actual deletion of the record
                        (when the record is just marked deleted, and the space occupied
                        by the data portion recovered if the file's prologue version
                        is 3), nor during the space recovery that is attempted when
                        there is insufficient room in the bucket to accomidate a new
                        record, or the increased size of an existing record. Therefore,
                        the routine RM$UPDATE_RRV must be modified, so that RRVS are
                        never created for deleted records in prologue 3 files, and so
                        that only deleted RRVs with no RRV pointers are created for
                        those deleted records in prologue 2 files which are in their
                        original buckets and require an RRV to preserve their ID from
                        being recycled.

                V03-003 TMK0001         Todd M. Katz            02-Jul-1982
                        Implement RMS cluster solution for next record positioning.
                        As the NRP cell has been eliminated and the next record
                        positioning context is now kept in the IRAB, refer to the
                        IRAB to obtain the RFA of the new/changed primary data record.
                        Also, as the module RM3NRP is disappearing, move the routines
                        RM$CODE_VBN and RM$SELECT_VBN to this module and make them
                        local routines.

                V03-002 MCN0012         Maria del C. Nasr       11-Jun-1982
                        Eliminate overhead at end of data bucket that was to be
                        used for duplicate continuation bucket processing.

                V03-001 SPR39795        L J Anderson            12-Mar-1982
                        In the case of a bucket split when run out of IDs, do
                        NOT update an RRV of a deleted record.  The deleted RRV
                        has the pointer space squished out, updating the RRV
                        results in a trashed bucket.

                V02-018 KBT0007         K B Thompson            15-Feb-1982
                        Add code to handle reclaimed bucket next-record-IDs and
                        add subtitles
```

```
;       0172  1 !
;       0173  1 !       V02-017 MCN0011         Maria del C. Nasr       28-May-1981
;       0174  1 !               More changes required for prologue 3 files.
;       0175  1 !
;       0176  1 !       V02-016 MCN0006         Maria del C. Nasr       16-Mar-1981
;       0177  1 !               Increase size of record identifier to a word in NRP, and
;       0178  1 !               other local structures.
;       0179  1 !
;       0180  1 !       V02-015 REFORMAT        C Saether       01-Aug-1980     22:38
;       0181  1 !
;       0182  1 !
;       0183  1 ! REVISION HISTORY:
;       0184  1 !
;       0185  1 !   Wendy Koenig,       28-SEP-78  9:11
;       0186  1 !   X0002 - SET RRV_ERR ON UPDATE ERROR, AND GO ON TO NEXT RRV
;       0187  1 !
;       0188  1 !   Wendy Koenig,       29-SEP-78  14:46
;       0189  1 !   X0003 - ADJUST POS_INS ON ANY SQUISH, NOT JUST IF BIG_SPLIT
;       0190  1 !
;       0191  1 !   Christian Saether,  12-OCT-78  12:20
;       0192  1 !   X0004 - do not release rrv buffer when in update mode
;       0193  1 !
;       0194  1 !   Wendy Koenig,       12-OCT-78  14:45
;       0195  1 !   X0005 - TAKE ALL THE NRP STUFF OUT OF HERE
;       0196  1 !
;       0197  1 !   Wendy Koenig,       17-OCT-78  15:40
;       0198  1 !   X0006 - CHANGE UPDATE_RRV FOR $UPDATE
;       0199  1 !
;       0200  1 !   Wendy Koenig,       24-OCT-78  14:03
;       0201  1 !   X0007 - MAKE CHANGES CAUSED BY SHARING CONVENTIONS
;       0202  1 !
;       0203  1 !   Christian Saether,  24-OCT-78  17:38
;       0204  1 !   X0008 - give UPDATE_RRV 1 more byte at end of buffer
;       0205  1 !
;       0206  1 !   Wendy Koenig,       26-OCT-78  11:29
;       0207  1 !   X0009 - GET RID OF DEFINITION OF IRC$B_RRV_ID WHICH IS NOW IN THE LIBRARY
;       0208  1 !
;       0209  1 !   Wendy Koenig,       31-OCT-78  14:09
;       0210  1 !   X0010 - FIX BIG, ONLY USE VBN_MID IF BIG_SPLIT
;       0211  1 !
;       0212  1 !   Christian Saether,  3-NOV-78  8:21
;       0213  1 !   X0011 - fix incorrect use of BDB$W_SIZE to BDB$W_NUMB
;       0214  1 !
;       0215  1 !   Wendy Koenig,       28-NOV-78  11:38
;       0216  1 !   X0012 - LOCK BUCKET WHEN UPDATING RRV'S
;       0217  1 !
;       0218  1 !   Christian Saether,  15-JAN-79  21:41
;       0219  1 !   X0013 - eliminate potential deadlock going for rrv's
;       0220  1 !
;       0221  1 !   Wendy Koenig,       26-JAN-79  9:20
;       0222  1 !   X0014 - GET RID OF SETTING VALID
;       0223  1 !
;       0224  1 !*****
;       0225  1
;       0226  1 LIBRARY 'RMSLIB:RMS';
;       0227  1
;       0228  1 REQUIRE 'RMSSRC:RMSIDXDEF';
```

```

! Define default PSECTS for code.
!
PSECT
    CODE = RM$RMS3(PSECT_ATTR),
    PLIT = RM$RMS3(PSECT_ATTR);

! Define some local MACROS.
!
MACRO
    IRC$L_RRV_VBN = 3,0,32,0 %,             ! location of RRV VBN in record
    IR3$L_RRV_VBN = 5,0,32,0 %,             ! new location in prologue 3 files

    ! The following macros which define the entries in the local table used for
    ! RRV updating, have been reordered to optimize prologue 3 file processing.
    ! Those fields that have not changed in size, have been placed up front, so
    ! that there are the least possible position variants.  The size of each
    ! RRV entry in the table is 10 bytes long for prologue 3 files, and 7 bytes
    ! for previous prologue versions.
    !
    TBL$W_FFB          = 0,0,16,0 %,                 ! stores table size
    TBL$B_NEW_VBN      = 0,0,8,0  %,                 ! new VBN index
    TBL$L_OLD_VBN      = 1,0,32,0 %,                 ! old VBN value
    TBL$B_NEW_ID       = 5,0,8,0  %,                 ! new record id
    TBL$W_NEW_ID       = 5,0,16,0 %,                 ! new record id (plg 3)
    TBL$B_OLD_ID       = 6,0,8,0  %,                 ! old record id
    TBL$W_OLD_ID       = 7,0,16,0 %,                 ! old record id (plg 3)
    TBL$B_FLAG         = 9,0,8,0  %,                 ! flag byte (prologue 3)
    TBL$V_RU_DELETE    = 9,0,1,0  %,                 ! record is RU_DELETEd

    FLG$V_POS_INS = 0,0,1,0 %,
    FLG$V_SPLIT_1 = 0,1,1,0 %,
    FLG$V_SPLIT_2 = 0,2,1,0 %,
    FLG$V_UPD_POS = 0,3,1,0 %,
    FLG$V_REC_DEL = 0,4,1,0 %;

! Linkages.
!
LINKAGE
    L_PRESERVE1,
    L_RABREG_4567,
    L_RABREG_457,
    L_RABREG_567,
    L_RABREG_67,
    L_RELEASE,

    ! Local linkages
    !
    RL$LINKAGE  = JSB() :
                  GLOBAL (R_IRAB),
    RL$SQUISH   = JSB (REGISTER = 3, REGISTER = 4)
                : GLOBAL (R_REC_ADDR);

! Forward Routines
!
FORWARD ROUTINE
    RM$SQUISH          : RL$SQUISH;
```

```
;  286          0350   1
;  287          0351   1 ! External Routines
;  288          0352   1 !
;  289          0353   1
;  290          0354   1 EXTERNAL ROUTINE
;  291          0355   1     RM$FIND_BY_ID          : RL$RABREG_567,
;  292          0356   1     RM$GETBKT              : RL$RABREG_457,
;  293          0357   1     RM$GETNEXT_REC         : RL$RABREG_67,
;  294          0358   1     RM$RECORD_ID           : RL$RABREG_67,
;  295          0359   1     RM$RECORD_VBN          : RL$PRESERVE1,
;  296          0360   1     RM$RELEASE             : RL$RELEASE ADDRESSING_MODE( GENERAL ),
;  297          0361   1     RM$RLSBKT              : RL$PRESERVE1;
;  298          0362   1
```

RM3
V04

```
1 %SBTTL 'RM$CODE_VBN'
1 ROUTINE RM$CODE_VBN (VBN) : RL$LINKAGE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 ! Converts the new VBN into a 1,2,3 to be stored away temporarily
1 ! NOTE: CODE_VBN and SELECT_VBN are complimentary routines.
1 !
1 ! CALLING SEQUENCE:
1 !       BSBW RM$CODE_VBN()
1 !
1 ! INPUT PARAMETERS:
1 !       the new VBN
1 !
1 ! IMPLICIT INPUTS:
1 !       IRAB -- VBN_RIGHT, VBN_MID, RFA_VBN
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       1,2,3
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         R_IRAB_STR;
2
3     RETURN (
3
3         SELECTONE .VBN OF
3             SET
3             [.IRAB[IRB$L_VBN_RIGHT]] : 1;
3             [.IRAB[IRB$L_VBN_MID]] : 2;
3             [.IRAB[IRB$L_RFA_VBN]] : 3;
2             TES);
2
1     END;                                            ! { end of CODE_VBN }

                                                       .TITLE  RM3RRV
                                                       .IDENT  \V04-000\

                                                       .EXTRN  RM$FIND_BY_ID, RM$GETBKT
                                                       .EXTRN  RM$GETNEXT_REC, RM$RECORD_ID
                                                       .EXTRN  RM$RECORD_VBN, RM$RELEASE
                                                       .EXTRN  RM$RLSBKT
```

```
                                                                   .PSECT  RM$RMS3,NOWRT,  GBL,  PIC,2

                              50       04    AE  D0 00000 RM$CODE_VBN:
                                                                   MOVL    VBN, R0                                       ; 0403
                        008C  C9             50  D1 00004          CMPL    R0, 140(IRAB)                                 ; 0405
                                             04  12 00009          BNEQ    1$                                            ;
                              50             01  D0 0000B          MOVL    #1, R0                                        ;
                                             05  0000E             RSB                                                   ;
                        0090  C9             50  D1 0000F 1$:      CMPL    R0, 144(IRAB)                                 ; 0406
                                             04  12 00014          BNEQ    2$                                            ;
                              50             02  D0 00016          MOVL    #2, R0                                        ;
                                             05  00019             RSB                                                   ;
                          70  A9             50  D1 0001A 2$:      CMPL    R0, 112(IRAB)                                 ; 0407
                                             04  13 0001E          BEQL    3$                                            ;
                              50             01  CE 00020          MNEGL   #1, R0                                        ;
                                             05  00023             RSB                                                   ;
                              50             03  D0 00024 3$:      MOVL    #3, R0                                        ;
                                             05  00027             RSB                                                   ; 0410

; Routine Size:  40 bytes,    Routine Base:  RM$RMS3 + 0000


;  348          0411  1
```

```
0412   1 %SBTTL 'RM$SELECT_VBN'
0413   1 ROUTINE RM$SELECT_VBN (VALUE, VBN) : RL$LINKAGE =
0414   1
0415   1 !++
0416   1 !
0417   1 ! FUNCTIONAL DESCRIPTION:
0418   1 !
0419   1 ! Converts the 0,1,2,3 which was stored in the RRV table into a relevant VBN.
0420   1 ! NOTE: CODE_VBN and SELECT_VBN are complimentary routines.
0421   1 !
0422   1 ! CALLING SEQUENCE:
0423   1 !       BSBW RM$SELECT_VBN()
0424   1 !
0425   1 ! INPUT PARAMETERS:
0426   1 !       VALUE -- 0,1,2,3 from the table entry
0427   1 !       VBN -- if value is 0, VBN is the value we want returned
0428   1 !
0429   1 ! IMPLICIT INPUTS:
0430   1 !       IRAB -- VBN_RIGHT, VBN_MID, RFA_VBN
0431   1 !
0432   1 ! OUTPUT PARAMETERS:
0433   1 !       NONE
0434   1 !
0435   1 ! IMPLICIT OUTPUTS:
0436   1 !       NONE
0437   1 !
0438   1 ! ROUTINE VALUE:
0439   1 !       the actual VBN associated w/ this entry
0440   1 !
0441   1 ! SIDE EFFECTS:
0442   1 !       NONE
0443   1 !
0444   1 !--
0445   1
0446   2     BEGIN
0447   2
0448   2     EXTERNAL REGISTER
0449   2         R_IRAB_STR;
0450   2
0451   3     RETURN (
0452   3
0453   3         CASE .VALUE FROM 0 TO 3 OF
0454   3             SET
0455   3             [0] : .VBN;
0456   3             [1] : .IRAB[IRB$L_VBN_RIGHT];
0457   3             [2] : .IRAB[IRB$L_VBN_MID];
0458   3             [3] : .IRAB[IRB$L_RFA_VBN];
0459   2             TES);
0460   2
0461   1     END;



        03              00        04   AE   CF 00000 RM$SELECT_VBN:
                                                              CASEL   VALUE, #0, #3                          ; 0453
        0019        0013        000D        0008      00005 1$:      .WORD    2$-1$,-                                    ;
                                                                               3$-1$,-                                    ;
                                                                               4$-1$,-                                    ;
                                                                               5$-1$                                      ;
                                  50          08    AE  D0 0000D 2$:      MOVL     VBN, R0                                   ; 0455
                                                        05 00011          RSB                                                ;
                                  50        008C    C9  D0 00012 3$:      MOVL     140(IRAB), R0                             ; 0456
                                                        05 00017          RSB                                                ;
                                  50        0090    C9  D0 00018 4$:      MOVL     144(IRAB), R0                             ; 0457
                                                        05 0001D          RSB                                                ;
                                  50          70    A9  D0 0001E 5$:      MOVL     112(IRAB), R0                             ; 0458
                                                        05 00022          RSB                                                ; 0461

; Routine Size:  35 bytes,    Routine Base:  RM$RMS3 + 0028


;  400         0462  1
;  402          0463  1 %SBTTL  'RM$SQISH'
;  403          0464  1 ROUTINE RM$SQUISH (EOB, SQUISH) : RL$SQUISH =
;  404          0465  1
;  405          0466  1 !++
;  406          0467  1 !
;  407          0468  1 ! FUNCTIONAL DESCRIPTION:
;  408          0469  1 !
;  409          0470  1 ! do the squishing w/o destroying all the registers
;  410          0471  1 !
;  411          0472  1 ! CALLING SEQUENCE:
;  412          0473  1 !       bsbw rm$squish (.eob, .squish);
;  413          0474  1 !
;  414          0475  1 ! INPUT PARAMETERS:
;  415          0476  1 !       eob -- address of end of data to be moved
;  416          0477  1 !       squish -- address of where data is to be moved into
;  417          0478  1 !
;  418          0479  1 ! IMPLICIT INPUTS:
;  419          0480  1 !       rec_addr -- address of beginning of data to be moved
;  420          0481  1 !
;  421          0482  1 ! OUTPUT PARAMETERS:
;  422          0483  1 !       NONE
;  423          0484  1 !
;  424          0485  1 ! IMPLICIT OUTPUTS:
;  425          0486  1 !       NONE
;  426          0487  1 !
;  427          0488  1 ! ROUTINE VALUE:
;  428          0489  1 !       rmssuc always
;  429          0490  1 !
;  430          0491  1 ! SIDE EFFECTS:
;  431          0492  1 !       some data records have been squished out
;  432          0493  1 !
;  433          0494  1 !--
;  434          0495  1
;  435          0496  2     BEGIN
;  436          0497  2
;  437          0498  2     EXTERNAL REGISTER
;  438          0499  2         R_REC_ADDR_STR;
;  439          0500  2
;  440          0501  2     CH$MOVE(.EOB - .REC_ADDR, .REC_ADDR, .SQUISH);
;  441          0502  2     RETURN RMSSUC();
;  442          0503  2
;  443          0504  1     END;                                     ! { end of routine }



                                        3C  BB 00000 RM$SQUISH:
                                                     PUSHR   #^M<R2,R3,R4,R5>                          ; 0464
                              53        56  C2 00002         SUBL2   REC_ADDR, R3                              ; 0501
               64             66        53  28 00005         MOVC3   R3, (REC_ADDR), (SQUISH)                  ;
                              50        01  D0 00009         MOVL    #1, R0                                    ; 0502
                                        3C  BA 0000C         POPR    #^M<R2,R3,R4,R5>                          ; 0504
                                            05 0000E         RSB                                               ;

; Routine Size:  15 bytes,    Routine Base.  RM$RMS3 + 004B
;  444         0505  1
```

```
1 %SBTTL  'RM$UPDATE_RRV'
1 GLOBAL ROUTINE RM$UPDATE_RRV : RL$RABREG_67 NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !     Create RRV's for records that moved out of this bucket w/o RRV's
1 !     and make a ta' e so that records that moved before can be updated later.
1 !     Do not make an entry in the table if the record has been deleted.
1 !
1 !     If a deleted record in its orginal bucket is encountered, make a RRV
1 !     for it if and only if the file's prologue version is not 3, and that RRV
1 !     is a deleted RRV without a pointer (to reserve the ID so it can not be
1 !     recycled).
1 !
1 ! CALLING SEQUENCE:
1 !     bsbw rm$update_rrv
1 !
1 ! INPUT PARAMETERS:
1 !     NONE
1 !
1 ! IMPLICIT INPUTS:
1 !     IRAB -- curbdb in irab describing the original bucket
1 !             nxtbdb describing the extra buffer being used to build the table
1 !     IDX_DFN - IDX$V_DUPKEYS
1 !     IFAB - IFB$B_PLG_VER
1 !
1 ! OUTPUT PARAMETERS:
1 !     NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !     NONE
1 !
1 ! ROUTINE VALUE:
1 !        nothing
1 !
1 ! SIDE EFFECTS:
1 !     The records that were moved out are physically deleted and rrv's are
1 !        built for all of them.
1 !     The bucket is marked dirty and valid.
1 !     Another buffer pointed to by nxtbdb is used to make a table to be used
1 !        to update rrv's in other buckets.
1 !     The split points except split itself and possibly pos_ins are destroyed.
1 !     Those two can still apply to the existing bucket
1 !     REC_ADDR is destroyed, but it was not an input.
1 !     Some convuluting stuff is done in the $update case, when there was an
1 !        original record.
1 !
1 !--
1
2    BEGIN
2
2    EXTERNAL REGISTER
2        COMMON_RAB_STR,
2        R_REC_ADDR_STR,
2        R_IDX_DFN_STR;
```

```
2
2       LOCAL
2           TABLE    : REF BBLOCK,
2           NXTID    : WORD,
2           REAL_END        : REF BBLOCK,
2           EOB      : REF BBLOCK,
2           SQUISH   : REF BBLOCK,
2           VBN,
2           POS_INS  : REF BBLOCK,
2           FLAG     : BLOCK [1],
2           RRV_VBN,
2           VBNT,
2           OLD_ID   : WORD;
2
2       GLOBAL REGISTER
2           R_BKT_ADDR_STR;
2
2       FLAG = 0;
2       TABLE = .BBLOCK[.IRAB[IRB$L_NXTBDB], BDB$L_ADDR] + 2;
2       BKT_ADDR = .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$L_ADDR];
2       REC_ADDR = .BKT_ADDR + .IRAB[IRB$W_SPLIT];
2       EOB = .BKT_ADDR[BKT$W_FREESPACE] + .BKT_ADDR;
2       REAL_END = .BKT_ADDR + .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$W_NUMB];
2
2       ! The real end of the bucket for prologue 3 files is different, since
2       ! there is some extra information at the end.  The checksum byte is
2       ! correctly accounted for, so add it back.
2       !
2       IF .IFAB[IFB$B_PLG_VER] EQLU PLG$C_VER_3
2       THEN
2           REAL_END = .REAL_END - BKT$C_DATBKTOVH + 1;
2
2       POS_INS = .BKT_ADDR + .IRAB[IRB$W_POS_INS];
2       SQUISH = .REC_ADDR;
2
2       ! Set Flag Position Insert, if intend on inserting the new record ( or
2       ! updating the record ) in the old left hand side bucket
2       !
2       IF .POS_INS LSSU .REC_ADDR
2       THEN
2           FLAG[FLG$V_POS_INS] = 1;
2
2       IF .POS_INS EQLU .REC_ADDR
2           AND
2           .IRAB[IRB$V_REC_W_LO]
2       THEN
2           FLAG[FLG$V_POS_INS] = 1;
2
2       ! Set up the starting vbn and the next-record-ID
2       !
2       IF .IRAB[IRB$V_BIG_SPLIT]
2       THEN
3           BEGIN
3           VBN = .IRAB [ IRB$L_VBN_MID ];
3           NXTID = .IRAB [ IRB$W_NID_MID ]
3           END
2       ELSE
```

```
3             BEGIN
3             VBN = .IRAB [ IRB$L_VBN_RIGHT ];
3             NXTID = .IRAB [ IRB$W_NID_RIGHT ]
2             END;
2
2       ! Skip through bucket, deciding where the RRV's for each record should be
2       ! put -- If in the old (left) bucket, put it at the end of that bucket.
2       ! If there is an RRV in another bucket, already; then it needs updating,
2       ! build an entry in the table.  Do not build an entry, if the record has
2       ! been deleted.
2       !
2
2       WHILE .REC_ADDR LEQU .EOB
2       DO
3           BEGIN
3
3           BUILTIN
3               AP;
3
3           LOCAL
3               DIFFERENCE  : WORD;
3
3           ! if rec_addr equal to the eob or we're at an rrv (virtual eob ),
3           ! we still need to do the update for a potential updated record at the
3           ! eob. but don't do it twice
3           !
3
3           IF .REC_ADDR EQLU .EOB
3               OR
3               .REC_ADDR[IRC$V_RRV]
3           THEN
3
3               IF .FLAG[FLG$V_POS_INS]
3                   OR
3                   NOT .IRAB[IRB$V_UPDATE]
3               THEN
3                   EXITLOOP;
3
3           ! If the record is deleted, then save this status in the FLAG byte.
3           !
3           IF .REC_ADDR NEQU .EOB
3               AND
3               .REC_ADDR[IRC$V_DELETED]
3           THEN
3               FLAG[FLG$V_REC_DEL] = 1
3           ELSE
3               FLAG[FLG$V_REC_DEL] = 0;
3
3           DIFFERENCE = .REC_ADDR - .BKT_ADDR;
3
3           ! if more than 1 new bucket, check to see if we've passed a split point
3           ! if so, the vbn and nxtid have to be changed
3           !
3
3           IF .IRAB[IRB$V_BIG_SPLIT]
3           THEN
4               BEGIN
```

```
4
4       IF .DIFFERENCE EQLU .IRAB[IRB$W_SPLIT_1]
4           AND
4           NOT .FLAG[FLG$V_SPLIT_1]
4       THEN
4
5           IF (.FLAG[FLG$V_POS_INS]
5               OR
5               NOT .IRAB[IRB$V_REC_W_LO])
4               OR
4               NOT .IRAB[IRB$V_UPDATE]
4           THEN
5               BEGIN
5               FLAG[FLG$V_SPLIT_1] = 1;
5
5               ! Use the RFA bucket
5               !
5               VBN = .IRAB [ IRB$L_RFA_VBN ];
5
5               ! If there is no RFA bucket then use the right bucket
5               ! else its ok to use the RFA bucket and next-record-ID
5               !
5               IF .VBN EQLU 0
5               THEN
6                   BEGIN
6                   VBN = .IRAB [ IRB$L_VBN_RIGHT ];
6                   NXTID = .IRAB [ IRB$W_NID_RIGHT ]
6                   END
5               ELSE
5                   NXTID = .IRAB [ IRB$W_RFA_NID ]
5
4               END;
4
4       IF .DIFFERENCE EQLU .IRAB[IRB$W_SPLIT_2]
4           AND
4           NOT .FLAG[FLG$V_SPLIT_2]
4       THEN
5           BEGIN
5
5           FLAG [ FLG$V_SPLIT_2 ] = 1;
5
5           VBN = .IRAB [ IRB$L_VBN_RIGHT ];
5           NXTID = .IRAB [ IRB$W_NID_RIGHT ]
5
4           END;
4
3       END;
3
3   ! if this is the pos for insert, and the record really and truly
3   ! belongs here, increment the nxtid but make sure that we can never
3   ! come back to pos_ins more than once if this is an update and the
3   ! record belonged in the middle bkt all by itself, set up vbn1  to
3   ! indicate such
3   !
3   VBN1 = .VBN;
3
3   IF .REC_ADDR EQLU .POS_INS
```

```
3           AND
3           NOT .FLAG[FLG$V_POS_INS]
3       THEN
4           BEGIN
4           FLAG[FLG$V_POS_INS] = 1;
4
4           IF .IRAB[IRB$V_UPDATE]
4           THEN
5               BEGIN
5               FLAG[FLG$V_UPD_POS] = 1;
5
5               IF .IRAB[IRB$V_BIG_SPLIT]
5                   AND
6                   (.IRAB[IRB$W_SPLIT] EQLU .IRAB[IRB$W_SPLIT_1])
5               THEN
6                   BEGIN
6                   FLAG[FLG$V_SPLIT_1] = 0;
6                   VBN1 = .IRAB[IRB$L_VBN_MID]
6                   END
5               END
5
4           ELSE
5               BEGIN
5               !
5               ! Ok, here's the scoop on what's going down here:
5               !
5               ! If this is the position for insert, AND the new
5               ! record doesn't go into a bucket all by itself
5               ! (i.e., a 3-bkt split), AND the new record doesn't
5               ! go into the old bucket, then skip an id to account
5               ! for the id taken up by the new record when it winds
5               ! up in the new bucket.
5               !
5               IF .IRAB[IRB$W_SPLIT] NEQU .IRAB[IRB$W_SPLIT_1]
5                   AND
5               NOT .IRAB[IRB$V_REC_W_LO]
5
5               THEN
5                   NXTID = .NXTID + 1
5               END
3           END;
3
3       AP = 3;
3
4       BEGIN
4
4       GLOBAL REGISTER
4           R_BDB;
4
4       IF .FLAG[FLG$V_UPD_POS]
4       THEN
4           RRV_VBN = .IRAB[IRB$L_PUTUP_VBN]
4       ELSE
4           RRV_VBN = RM$RECORD_VBN();
3       END;
3
3       ! if the VBN's are equal, then this record has never moved and, thus
```

```
3          ! it needs an RRV; otherwise, it has an RRV elsewhere. NOTE that there
3          ! is no need to create an RRV for this record (even if the the VBNs
3          ! are equal) if the record is deleted and the file is a prologue 3
3          ! file.
3          !
3          IF .RRV_VBN EQLU .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$L_VBN]
3              AND
5             (NOT (.IFAB[IFB$B_PLG_VER] GEQU PLG$C_VER_3
5                              AND
5                            .FLAG[FLG$V_REC_DEL])
4                     OR
4                   .FLAG[FLG$V_UPD_POS])
3          THEN
4              BEGIN
4
4              LOCAL
4                  RRV_SIZE;
4
4              IF .FLAG[FLG$V_UPD_POS]
4              THEN
4                  OLD_ID = .IRAB[IRB$W_PUTUP_ID]
4              ELSE
4                  OLD_ID = RM$RECORD_ID();
4
4              IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
4              THEN
4                  IF NOT .FLAG[FLG$V_REC_DEL]
4                  THEN
4                      RRV_SIZE = 7
4                  ELSE
4                      RRV_SIZE = 2
4              ELSE
4                  RRV_SIZE = 9;
4
4              ! if there is not enough physical room at the end of the bucket to
4              ! build an rrv, make enough
4              !
4              IF (.EOB + .RRV_SIZE) GEQU .REAL_END
4              THEN
5                  BEGIN
5
5                  IF NOT .FLAG[FLG$V_UPD_POS]
5                  THEN
5                      RM$GETNEXT_REC();
5
5                  RM$SQUISH(.EOB, .SQUISH);
5                  EOB = .EOB - (.REC_ADDR - .SQUISH);
5
5                  ! unfortunately, if we squish records out, we also have to
5                  ! update all the pointers to the bucket
5                  !
5
5                  IF .IRAB[IRB$V_BIG_SPLIT]
5                  THEN
6                      BEGIN
6
6                      IF .SQUISH LEQU .BKT_ADDR + .IRAB[IRB$W_SPLIT_1]
```

```
6                       THEN
7                           BEGIN

7                           IF .BKT_ADDR + .IRAB[IRB$W_SPLIT_1] LEQU .REC_ADDR
7                           THEN
7                               IRAB[IRB$W_SPLIT_1] = .SQUISH - .BKT_ADDR
7                           ELSE
7                               IRAB[IRB$W_SPLIT_1] = .IRAB[IRB$W_SPLIT_1] -
8                               (.REC_ADDR - .SQUISH)
6                           END;
6
6                       IF .SQUISH LEQU .BKT_ADDR + .IRAB[IRB$W_SPLIT_2]
6                       THEN
7                           BEGIN
7
7                           IF .BKT_ADDR + .IRAB[IRB$W_SPLIT_2] LEQU .REC_ADDR
7                           THEN
7                               IRAB[IRB$W_SPLIT_2] = .SQUISH - .BKT_ADDR
7                           ELSE
7                               IRAB[IRB$W_SPLIT_2] = .IRAB[IRB$W_SPLIT_2] -
8                               (.REC_ADDR - .SQUISH)
6                           END;
6
5                       END;
5
5                   IF .SQUISH LEQU .POS_INS
5                   THEN
6                       BEGIN
6
6                       IF .POS_INS LEQU .REC_ADDR
6                       THEN
6                           POS_INS = .SQUISH
6                       ELSE
7                           POS_INS = .POS_INS - (.REC_ADDR - .SQUISH)
5                       END;
5
5                   REC_ADDR = .SQUISH;
5                   END
5
5               ! Else we do not have to squish a record out.
5               !
4               ELSE
4                   IF NOT .FLAG[FLG$V_UPD_POS]
4                   THEN
4                       RM$GETNEXT_REC();
4
4               ! Build the RRV at the end of the bucket and update EOB
4               !
4               EOB[IRC$B_CONTROL] = 0;
4               EOB[IRC$V_RRV] = 1;
4
4               IF .IFAB[IFB$B_FLG_VER] LSSU PLG$C_VER_3
4               THEN
4
4                   ! If the record is deleted and the file is not a prologue 3
4                   ! file then created a two-byte deleted RRV for the record.
4                   !
```

```
4               IF .FLAG[FLG$V_REC_DEL]
4               THEN
5                   BEGIN
5                   EOB[IRC$V_NOPTRSZ] = 1;
5                   EOB[IRC$V_DELETED] = 1;
5                   EOB[IRC$B_ID] = .OLD_ID;
5                   EOB = .EOB + 2;
5                   END
4               ELSE
5                   BEGIN
5                   EOB[IRC$V_PTRSZ] = 2;
5                   EOB[IRC$B_ID] = .OLD_ID;
5                   EOB[IRC$B_RRV_ID] = .NXTID;
5                   EOB[IRC$L_RRV_VBN] = .VBN1;
5                   EOB = .EOB  + $BYTEOFFSET(IRC$L_RRV_VBN)
5                               + $BYTESIZE(IRC$L_RRV_VBN);
5                   END
4           ELSE
5               BEGIN
5               EOB[IRC$V_PTRSZ] = 2;
5               EOB[IRC$W_ID] = .OLD_ID;
5               EOB[IRC$W_RRV_ID] = .NXTID;
5               EOB[IR3$L_RRV_VBN] = .VBN1;
5               EOB = .EOB + $BYTEOFFSET(IR3$L_RRV_VBN)
5                          + $BYTESIZE(IR3$L_RRV_VBN);
4               END;
4           END
4
4       ! the record has moved before, so make an entry in the table so we can
4       ! update the record's old RRV, later.  Make an entry only if the record
4       ! is present (ie, do not update deleted RRV's).  The only time there will
4       ! be a deleted record in the middle of the bucket, is if this split is
4       ! happening because of no more id's available (not because of lack of
4       ! space).  In this case, the routine to squish the deleted records out
4       ! of the bucket is not called, as space is not the problem.
4       !
4       !
3       ELSE
3           IF NOT .FLAG[FLG$V_REC_DEL]
3           THEN
4               BEGIN
4               TABLE[TBL$B_NEW_VBN] = RM$CODE_VBN(.VBN1);
4               TABLE[TBL$L_OLD_VBN] = .RRV_VBN;
4
4               IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
4               THEN
5                   BEGIN
5                   TABLE[TBL$B_NEW_ID] = .NXTID;
5
5                   IF .FLAG[FLG$V_UPD_POS]
5                   THEN
5                       TABLE[TBL$B_OLD_ID] = .IRAB[IRB$W_PUTUP_ID]
5                   ELSE
5                       TABLE[TBL$B_OLD_ID] = .REC_ADDR[IRC$B_RRV_ID];
5
5                   TABLE = .TABLE + 7;
5                   END
```

```
4               ELSE
5                   BEGIN
5                   TABLE[TBL$W_NEW_ID] = .NXTID;
5
5                   IF .FLAG[FLG$V_UPD_POS]
5                   THEN
5                       TABLE[TBL$W_OLD_ID] = .IRAB[IRB$W_PUTUP_ID]
5                   ELSE
6                       BEGIN
6                       TABLE[TBL$W_OLD_ID] = .REC_ADDR[IRC$W_RRV_ID];
6
6                       ! If the current record was deleted within a Recovery
6                       ! Unit, then save this information in the flag byte
6                       ! of the table entry.
6                       !
6                       IF .REC_ADDR[IRC$V_RU_DELETE]
6                       THEN
6                           TABLE[TBL$V_RU_DELETE] = 1;
5                       END;
5
5                   TABLE = .TABLE + 10;
4                   END;
4
4               IF NOT .FLAG[FLG$V_UPD_POS]
4               THEN
4                   RM$GETNEXT_REC()
4
4               END             ! end of else record has moved before !
4
4           ! Else the current record is a deleted record, then just get the next
4           ! record.  (Do not need to check FLG$V_UPD_POS, because on a bucket
4           ! split because of no more id's available, it was on an insert oper-
4           ! ation, not an update).
4           !
3           ELSE
3               RM$GETNEXT_REC();
3
3       ! bump the nxtid
3       !
3       NXTID = .NXTID + 1;
3
3       ! clear the "at pos_for_insert in update mode" flag
3       !
3       FLAG[FLG$V_UPD_POS] = 0;
2       END;                                    ! { end of while loop }
2
2   ! if there still are records that need to be squashed out, do it
2   !
2
2   IF .SQUISH NEQU .REC_ADDR
2   THEN
3       BEGIN
3       RM$SQUISH(.EOB, .SQUISH);
3       EOB = .EOB - (.REC_ADDR - .SQUISH);
3       REC_ADDR = .SQUISH;
2       END;
2
```

```
;  959          1019  2        ! update the freespace word
;  960          1020  2        !
;  961          1021  2        BKT_ADDR[BKT$W_FREESPACE] = .EOB - .BKT_ADDR;
;  962          1022  2
;  963          1023  2        ! mark the end of the table in its first word for future reference
;  964          1024  2        !
;  965          1025  3        BEGIN
;  966          1026  3
;  967          1027  3        LOCAL
;  968          1028  3            BEG_TABLE       : REF BBLOCK;
;  969          1029  3
;  970          1030  3        BEG_TABLE = .BBLOCK[.IRAB[IRB$L_NXTBDB], BDB$L_ADDR];
;  971          1031  3        BEG_TABLE[TBL$W_FFB] = .TABLE - .BEG_TABLE
;  972          1032  2        END;
;  973          1033  2        RETURN;
;  974          1034  2
;  975          1035  1        END;                                      ! { end of routine }



                                        3C  BB 00000 RM$UPDATE_RRV::
                                                              PUSHR   #^M<R2,R3,R4,R5>                           ; 0507
                   5E                   1C  C2 00002          SUBL2   #28, SP
                                        7E  D4 00005          CLRL    FLAG                                       ; 0580
                   50         3C        A9  D0 00007          MOVL    60(IRAB), R0                               ; 0581
       52     18   A0                   02  C1 0000B          ADDL3   #2, 24(R0), TABLE                          ;
                   50         20        A9  D0 00010          MOVL    32(IRAB), R0                               ; 0582
                   55         18        A0  D0 00014          MOVL    24(R0), BKT_ADDR
                   56         4A        A9  3C 00018          MOVZWL  74(IRAB), REC_ADDR                         ; 0583
                   56                   55  C0 0001C          ADDL2   BKT_ADDR, REC_ADDR                         ;
                   53         04        A5  3C 0001F          MOVZWL  4(BKT_ADDR), EOB                           ; 0584
                   53                   55  C0 00023          ADDL2   BKT_ADDR, EOB                              ;
                   51         14        A0  3C 00026          MOVZWL  20(R0), R1                                 ; 0585
                                      6145  9F 0002A          PUSHAB  (R1)[BKT_ADDR]                             ;
                   03       00B7        CA  91 0002D          CMPB    183(IFAB), #3                              ; 0591
                                        02  12 00032          BNEQ    1$                                         ;
                                        6E  D7 00034          DECL    REAL_END                                   ; 0593
                   50         48        A9  3C 00036 1$:      MOVZWL  72(IRAB), R0                               ; 0595
  10   AE                               55  50  C1 0003A      ADDL3   R0, BKT_ADDR, POS_INS
              08   AE                   56  D0 0003F          MOVL    REC_ADDR, SQUISH                           ; 0596
                   56         10        AE  D1 00043          CMPL    POS_INS, REC_ADDR                          ; 0601
                                        04  1E 00047          BGEQU   2$                                         ;
              04   AE                   01  88 00049          BISB2   #1, FLAG                                   ; 0603
                   56         10        AE  D1 0004D 2$:      CMPL    POS_INS, REC_ADDR                          ; 0605
                                        09  12 00051          BNEQ    3$                                         ;
       04     44   A9                   03  E1 00053          BBC     #3, 68(IRAB), 3$                           ; 0607
              04   AE                   01  88 00058          BISB2   #1, FLAG                                   ; 0609
       0E     44   A9                   02  E1 0005C 3$:      BBC     #2, 68(IRAB), 4$                           ; 0613
              14   AE       0090        C9  D0 00061          MOVL    144(IRAB), VBN                             ; 0616
              0C   AE       00A2        C9  B0 00067          MOVW    162(IRAB), NXTID                           ; 0617
                                        0C  11 0006D          BRB     5$                                         ;
              14   AE       008C        C9  D0 0006F 4$:      MOVL    140(IRAB), VBN                             ; 0621
              0C   AE       00A0        C9  B0 00075          MOVW    160(IRAB), NXTID                           ; 0622
                   53                   56  D1 0007B 5$:      CMPL    REC_ADDR, EOB                              ; 0632
                                        03  1B 0007E          BLEQU   7$                                         ;
                                      0254  31 00080 6$:     BRW      50$                                         ;        :
                                        04  13 00083 7$:     BEQL     8$                                          ; 0647   :
     09             66                  03  E1 00085         BBC      #3, (REC_ADDR), 9$                          ; 0649   :
                    F3          04      AE  E8 00089 8$:     BLBS     FLAG, 6$                                    ; 0652   :
     EE     06      A9                  03  E1 0008D         BBC      #3, 6(IRAB), 6$                             ; 0654   :
                                        0A  13 00092 9$:     BEQL     10$                                         ; 0660   :
     06             66                  02  E1 00094         BBC      #2, (REC_ADDR), 10$                         ; 0662   :
            04      AE                  10  88 00098         BISB2    #16, FLAG                                   ; 0664   :
                                        04  11 0009C         BRB      11$                                         ;        :
            04      AE                  10  8A 0009E 10$:    BICB2    #16, FLAG                                   ; 0666   :
     50             56                  55  A3 000A2 11$:    SUBW3    BKT_ADDR, REC_ADDR, DIFFERENCE              ; 0668   :
     53     44      A9                  02  E1 000A6         BBC      #2, 68(IRAB), 15$                           ; 0674   :
            4C      A9                  50  B1 000AB         CMPW     DIFFERENCE, 76(IRAB)                        ; 0678   :
                                        32  12 000AF         BNEQ     14$                                         ;        :
     2D     04      AE                  01  E0 000B1         BBS      #1, FLAG, 14$                               ; 0680   :
                    0A          04      AE  E8 000B6         BLBS     FLAG, 12$                                   ; 0683   :
     05     44      A9                  03  E1 000BA         BBC      #3, 68(IRAB), 12$                           ; 0685   :
     1F     06      A9                  03  E0 000BF         BBS      #3, 6(IRAB), 14$                            ; 0687   :
            04      AE                  02  88 000C4 12$:    BISB2    #2, FLAG                                    ; 0690   :
            14      AE          70      A9  D0 000C8         MOVL     112(IRAB), VBN                              ; 0694   :
                                        0E  12 000CD         BNEQ     13$                                         ; 0699   :
            14      AE        008C      C9  D0 000CF         MOVL     140(IRAB), VBN                              ; 0702   :
            0C      AE        00A0      C9  B0 000D5         MOVW     160(IRAB), NXTID                            ; 0703   :
                                        06  11 000DB         BRB      14$                                         ;        :
            0C      AE        00A4      C9  B0 000DD 13$:    MOVW     164(IRAB), NXTID                            ; 0706   :
            4E      A9                  50  B1 000E3 14$:    CMPW     DIFFERENCE, 78(IRAB)                        ; 0710   :
                                        15  12 000E7         BNEQ     15$                                         ;        :
     10     04      AE                  02  E0 000E9         BBS      #2, FLAG, 15$                               ; 0712   :
            04      AE                  04  88 000EE         BISB2    #4, FLAG                                    ; 0716   :
            14      AE        008C      C9  D0 000F2         MOVL     140(IRAB), VBN                              ; 0718   :
            0C      AE        00A0      C9  B0 000F8         MOVW     160(IRAB), NXTID                            ; 0719   :
            18      AE          14      AE  D0 000FE 15$:    MOVL     VBN, VBN1                                   ; 0731   :
            10      AE                  56  D1 00103         CMPL     REC_ADDR, POS_INS                           ; 0733   :
                                        38  12 00107         BNEQ     17$                                         ;        :
                    34          04      AE  E8 00109         BLBS     FLAG, 17$                                   ; 0735   :
            04      AE                  01  88 0010D         BISB2    #1, FLAG                                    ; 0738   :
     1C     06      A9                  03  E1 00111         BBC      #3, 6(IRAB), 16$                            ; 0740   :
            04      AE                  08  88 00116         BISB2    #8, FLAG                                    ; 0743   :
     22     44      A9                  02  E1 0011A         BBC      #2, 68(IRAB), 17$                           ; 0745   :
            4C      A9          4A      A9  B1 0011F         CMPW     74(IRAB), 76(IRAB)                          ; 0747   :
                                        1B  12 00124         BNEQ     17$                                         ;        :
            04      AE                  02  8A 00126         BICB2    #2, FLAG                                    ; 0750   :
            18      AE        0090      C9  D0 0012A         MOVL     144(IRAB), VBN1                             ; 0751   :
                                        0F  11 00130         BRB      17$                                         ; 0745   :
            4C      A9          4A      A9  B1 00132 16$:    CMPW     74(IRAB), 76(IRAB)                          ; 0767   :
                                        08  13 00137         BEQL     17$                                         ;        :
     03     44      A9                  03  E0 00139         BBS      #3, 68(IRAB), 17$                           ; 0769   :
                                0C      AE  B6 0013E         INCW     NXTID                                       ; 0772   :
                    5C                  03  D0 00141 17$:    MOVL     #3, AP                                      ; 0776   :
     07     04      AE                  03  E1 00144         BBC      #3, FLAG, 18$                               ; 0783   :
            20      AE          78      A9  D0 00149         MOVL     120(IRAB), RRV_VBN                          ; 0785   :
                                        07  11 0014E         BRB      19$                                         ;        :
                                      0000G 30 00150 18$:    BSBW     RM$RECORD_VBN                               ; 0787   :
            20      AE                  50  D0 00153         MOVL     R0, RRV_VBN                                 ;        :
                    50          20      A9  D0 00157 19$:    MOVL     32(IRAB), R0                                ; 0796   :
            1C      A0          20      AE  D1 0015B         CMPL     RRV_VBN, 28(R0)                             ;        :
                                        03  13 00160         BEQL     21$                                         ;        :
```

```
                                                         G 10
RM3RRV                                          16-Sep-1984 02:00:47    VAX-11 Bliss-32 V4.0-742                Page  24
V04-000          RM$UPDATE_RRV                  14-Sep-1984 13:01:39    [ISK$VMSMASTER:[RMS.SRC]RM3RRV.B32;1      (5)

                                      0105 31 00162 20$: BRW     41$                                  ;
                          03    00B7    CA 91 00165 21$: CMPB    183(IFAB), #3                        ; 0798
                                        0A 1F 0016A      BLSSU   22$                                  ;
              05    04    AE            04 E1 0016C      BBC     #4, FLAG, 22$                        ; 0800
              EC    04    AE            03 E1 00171      BBC     #3, FLAG, 20$                        ; 0802
              08    04    AE            03 E1 00176 22$: BBC     #3, FLAG, 23$                        ; 0809
              1C          AE    0080    C9 B0 0017B      MOVW    128(IRAB), OLD_ID                    ; 0811
                                        07 11 00181      BRB     24$                                  ;
                                     0000G 30 00183 23$: BSBW    RM$RECORD_ID                         ; 0813
                    1C    AE            50 B0 00186      MOVW    R0, OLD_ID                           ;
                          03    00B7    CA 91 0018A 24$: CMPB    183(IFAB), #3                        ; 0815
                                        0F 1E 0018F      BGEQU   26$                                  ;
              05    04    AE            04 E0 00191      BBS     #4, FLAG, 25$                        ; 0817
                          50            07 D0 00196      MOVL    #7, RRV_SIZE                         ; 0819
                                        08 11 00199      BRB     27$                                  ;
                          50            02 D0 0019B 25$: MOVL    #2, RRV_SIZE                         ; 0821
                                        03 11 0019E      BRB     27$                                  ; 0817
                          50            09 D0 001A0 26$: MOVL    #9, RRV_SIZE                         ; 0823
  51    04    AE          01            03 EF 001A3 27$: EXTZV   #3, #1, FLAG, R1                     ; 0832
                          51            51 D2 001A9      MCOML   R1, R1                               ;
                          50            53 C0 001AC      ADDL2   EOB, R0                              ; 0828
                          6E            50 D1 001AF      CMPL    R0, REAL_END                         ;
                                        74 1F 001B2      BLSSU   35$                                  ;
                          03            51 E9 001B4      BLBC    R1, 28$                              ; 0832
                                     0000G 30 001B7      BSBW    RM$GETNEXT_REC                       ; 0834
                          54      08    AE D0 001BA 28$: MOVL    SQUISH, R4                           ; 0836
                                      FE30 30 001BE      BSBW    RM$SQUISH                            ;
              54    08    AE            56 C3 001C1      SUBL3   REC_ADDR, SQUISH, R4                 ; 0837
                          53            54 C0 001C6      ADDL2   R4, EOB                              ;
              3C    44    A9            02 E1 001C9      BBC     #2, 68(IRAB), 32$                    ; 0843
                          50      4C    A9 3C 001CE      MOVZWL  76(IRAB), R0                         ; 0847
                          50            55 C0 001D2      ADDL2   BKT_ADDR, R0                         ;
                          50      08    AE D1 001D5      CMPL    SQUISH, R0                           ;
                                        11 1A 001D9      BGTRU   30$                                  ;
                          56            50 D1 001DB      CMPL    R0, REC_ADDR                         ; 0851
                                        08 1A 001DE      BGTRU   29$                                  ;
        4C    A9    08    AE            55 A3 001E0      SUBW3   BKT_ADDR, SQUISH, 76(IRAB)           ; 0853
                                        04 11 001E6      BRB     30$                                  ;
                    4C    A9            54 A0 001E8 29$: ADDW2   R4, 76(IRAB)                         ; 0856
                          50      4E    A9 3C 001EC 30$: MOVZWL  78(IRAB), R0                         ; 0859
                          50            55 C0 001F0      ADDL2   BKT_ADDR, R0                         ;
                          50      08    AE D1 001F3      CMPL    SQUISH, R0                           ;
                                        11 1A 001F7      BGTRU   32$                                  ;
                          56            50 D1 001F9      CMPL    R0, REC_ADDR                         ; 0863
                                        08 1A 001FC      BGTRU   31$                                  ;
        4E    A9    08    AE            55 A3 001FE      SUBW3   BKT_ADDR, SQUISH, 78(IRAB)           ; 0865
                                        04 11 00204      BRB     32$                                  ;
                    4E    A9            54 A0 00206 31$: ADDW2   R4, 78(IRAB)                         ; 0868
                    10    AE      08    AE D1 0020A 32$: CMPL    SQUISH, POS_INS                      ; 0873
                                        11 1A 0020F      BGTRU   34$                                  ;
                          56      10    AE D1 00211      CMPL    POS_INS, REC_ADDR                    ; 0877
                                        07 1A 00215      BGTRU   33$                                  ;
                    10    AE      08    AE D0 00217      MOVL    SQUISH, POS_INS                      ; 0879
                                        04 11 0021C      BRB     34$                                  ;
                    10    AE            54 C0 0021E 33$: ADDL2   R4, POS_INS                          ; 0881
                          56      08    AE D0 00222 34$: MOVL    SQUISH, REC_ADDR                     ; 0884
                                        06 11 00226      BRB     36$                                  ; 0828
```

```
RM3RRV                                                        H 10
V04-000          RM$UPDATE_RRV                       16-Sep-1984 02:00:47   VAX-11 Bliss-32 V4.0-742              Page  25
                                                     14-Sep-1984 13:01:39   DISK$VMSMASTER:[RMS.SRC]RM3RRV.B32;1   (5)

                     03                51 E9 00228 35$:  BLBC    R1, 36$                              ; 0890
                                    0000G 30 0022B       BSBW    RM$GETNEXT_REC                       ; 0892
                                       63 94 0022E 36$:  CLRB    (EOB)                                ; 0896
                     63                08 88 00230       BISB2   #8, (EOB)                            ; 0897
                     03       00B7     CA 91 00233       CMPB    183(IFAB), #3                        ; 0899
                                       1D 1E 00238       BGEQU   38$                                  ;
        09     04    AE                04 E1 0023A       BBC     #4, FLAG, 37$                        ; 0905
                     83                14 88 0023F       BISB2   #20, (EOB)+                          ; 0909
                     83         1C     AE 90 00242       MOVB    OLD_ID, (EOB)+                       ; 0910
                                       20 11 00246       BRB     40$                                  ; 0905
83      02           00                02 F0 00248 37$:  INSV    #2, #0, #2, (EOB)+                   ; 0915
                     83         1C     AE 90 0024D       MOVB    OLD_ID, (EOB)+                       ; 0916
                     83         0C     AE 90 00251       MOVB    NXTID, (EOB)+                        ; 0917
                                       0D 11 00255       BRB     39$                                  ; 0918
83      02           00                02 F0 00257 38$:  INSV    #2, #0, #2, (EOB)+                   ; 0924
                     83         1C     AE B0 0025C       MOVW    OLD_ID, (EOB)+                       ; 0925
                     83         0C     AE B0 00260       MOVW    NXTID, (EOB)+                        ; 0926
                     83         18     AE D0 00264 39$:  MOVL    VBN1, (EOB)+                         ; 0927
                                       63 11 00268 40$:  BRB     49$                                  ; 0796
        5B     04    AE                04 E0 0026A 41$:  BBS     #4, FLAG, 48$                        ; 0943
                                18     AE DD 0026F       PUSHL   VBN1                                 ; 0946
                                     FD31 30 00272       BSBW    RM$CODE_VBN                          ;
                     5E                04 C0 00275       ADDL2   #4, SP                               ;
                     62                50 90 00278       MOVB    R0, (TABLE)                          ;
               01    A2         20     AE D0 0027B       MOVL    RRV_VBN, 1(TABLE)                    ; 0947
                     03       00B7     CA 91 00280       CMPB    183(IFAB), #3                        ; 0949
                                       1C 1E 00285       BGEQU   44$                                  ;
               05    A2         0C     AE 90 00287       MOVB    NXTID, 5(TABLE)                      ; 0952
        08     04    AE                03 E1 0028C       BBC     #3, FLAG, 42$                        ; 0956
               06    A2       0080     C9 90 00291       MOVB    128(IRAB), 6(TABLE)                  ;
                                       05 11 00297       BRB     43$                                  ;
               06    A2         02     A6 90 00299 42$:  MOVB    2(REC_ADDR), 6(TABLE)                ; 0958
                     52                07 C0 0029E 43$:  ADDL2   #7, TABLE                            ; 0960
                                       22 11 002A1       BRB     47$                                  ; 0949
               05    A2         0C     AE B0 002A3 44$:  MOVW    NXTID, 5(TABLE)                      ; 0964
        08     04    AE                03 E1 002A8       BBC     #3, FLAG, 45$                        ; 0968
               07    A2       0080     C9 B0 002AD       MOVW    128(IRAB), 7(TABLE)                  ;
                                       0D 11 002B3       BRB     46$                                  ;
               07    A2         03     A6 B0 002B5 45$:  MOVW    3(REC_ADDR), 7(TABLE)                ; 0971
        04           66                05 E1 002BA       BBC     #5, (REC_ADDR), 46$                  ; 0977
               09    A2                01 88 002BE       BISB2   #1, 9(TABLE)                         ; 0979
                     52                0A C0 002C2 46$:  ADDL2   #10, TABLE                           ; 0982
        03     04    AE                03 E0 002C5 47$:  BBS     #3, FLAG, 49$                        ; 0985
                                    0000G 30 002CA 48$:  BSBW    RM$GETNEXT_REC                       ; 0997
                                0C     AE B6 002CD 49$:  INCW    NXTID                                ; 1001
               04    AE                08 8A 002D0       BICB2   #8, FLAG                             ; 1005
                                     FDA4 31 002D4       BRW     5$                                   ; 0632
                     56         08     AE D1 002D7 50$:  CMPL    SQUISH, REC_ADDR                     ; 1011
                                       13 13 002DB       BEQL    51$                                  ;
                     54         08     AE D0 002DD       MOVL    SQUISH, R4                           ; 1014
                                     FD0D 30 002E1       BSBW    RM$SQUISH                            ;
        54     08    AE                56 C3 002E4       SUBL3   REC_ADDR, SQUISH, R4                 ; 1015
                     53                54 C0 002E9       ADDL2   R4, EOB                              ;
                     56         08     AE D0 002EC       MOVL    SQUISH, REC_ADDR                     ; 1016
    04  A5           53                55 A3 002F0 51$:  SUBW3   BKT_ADDR, EOB, 4(BKT_ADDR)           ; 1021
                     50         3C     A9 D0 002F5       MOVL    60(IRAB), R0                         ; 1030
                     50         18     A0 D0 002F9       MOVL    24(R0), BEG_TABLE                    ;
```

```
                       60          52           50  A3 002FD          SUBW3    BEG_TABLE, TABLE, (BEG_TABLE)              ; 1031
                                   5E           24  C0 00301          ADDL2    #36, SP                                    ; 1035
                                                3C  BA 00304          POPR     #^M<R2,R3,R4,R5>                           ;
                                                    05 00306          RSB                                                 ;

; Routine Size:  775 bytes,    Routine Base:  RMS$RMS3 + 005A


;  976          1036  1
```

```
1 %SBTTL  'RM$UPDATE_RRV_2'
1 GLOBAL ROUTINE RM$UPDATE_RRV_2 : RL$RABREG_4567 NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1    Update the rrv's from other buckets.  Return with IRAB[IRB$V_RRV_ERR] set,
1    if an error occurs during the update if it will cause the bucket to be trashed.
1
1 ! CALLING SEQUENCE:
1 !       bsbw rm$update_2
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       irab --
1 !           nxtbdb -- referring to table of rrv's
1 !           vbn_right, vbn_mid, rfa_vbn
1 !           abovelckd - set when level 1 was locked coming down tree
1 !       rab -- to store stv in
1 !       idx_dfn, IFAB, impure area, for rm$getbkt
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       nxtbdb is released and cleared
1 !       rrv_err is set in the irab on any error
1 !
1 ! ROUTINE VALUE:
1 !       none -- rrv_err is set in the irab on any error
1 !               and the stv contains the actual status
1 !
1 ! SIDE EFFECTS:
1 !       rec_addr, ap, and bkt_addr are destroyed
1 !       nxtbdb is released and cleared
1 !       many buckets may be accessed and written out
1 !
1 !--
1
2    BEGIN
2
2    EXTERNAL REGISTER
2        COMMON_IO_STR,
2        R_REC_ADDR_STR,
2        COMMON_RAB_STR,
2        R_IDX_DFN_STR;
2
2    LOCAL
2        TABLE    : REF BBLOCK,
2        EOT;
2
2    LABEL
2        INNER,
2        INNERMOST,
2           BLK,
2           BLOCK;
2
2 BLOCK :
3       BEGIN
3
3       LOCAL
3           ENTRY_SIZE;
3
3       TABLE = .BBLOCK[.IRAB[IRB$L_NXTBDB], BDB$L_ADDR];
3       EOT = .TABLE + .TABLE[TBL$W_FFB];
3       TABLE = .TABLE + 2;
3
3       IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
3       THEN
3           ENTRY_SIZE = 7
3       ELSE
3           ENTRY_SIZE = 10;
3
3       ! while there are still entries in the table, update each rrv individually
3       !
3
3       WHILE .TABLE LSSU .EOT
3       DO
4           BEGIN
4
4           ! if the table entry has already been taken care of, its vbn has
4           ! been cleared, so ignore it.
4           !
4
4           IF .TABLE[TBL$L_OLD_VBN] NEQ 0
4           THEN
4 INNER :
5               BEGIN
5
5               ! get the bucket to be updated
5               !
5 BLK :
6               BEGIN
6
6               LOCAL
6                   ST,
6                   SIZE;
6
6               SIZE = .IDX_DFN[IDX$B_DATBKTSZ]*512;
6               IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
6
6               ! if level above locked we must read the bucket with nowait to
6               ! avoid potential deadlock situation
6               !
6
6               IF .IRAB[IRB$V_ABOVELCKD]
6               THEN
6                   BBLOCK[IRAB[IRB$B_CACHEFLGS], CSH$V_NOWAIT] = 1;
6
6               ST = RM$GETBKT(.TABLE[TBL$L_OLD_VBN], .SIZE);
6
```

```
6                IF .ST
6                THEN
6                    LEAVE BLK;
6
7                IF .ST<0, 16> EQL RMSERR(RLK)
6                THEN
7                    BEGIN
7
7                    ! we got a record lock error on the bucket so clear the flag
7                    ! and release the level 1 bucket to remove the deadlock
7                    ! potential
7                    !
7                    IRAB[IRB$V_ABOVELCKD] = 0;
7                    BDB = .IRAB[IRB$L_LOCK_BDB];
7                    IRAB[IRB$L_LOCK_BDB] = 0;
7                    RM$RLSBKT(0);
7
7                    ! re-read the bucket we want and wait for it this time
7                    !
7                    IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
7                    ST = RM$GETBKT(.TABLE[TBL$L_OLD_VBN], .SIZE);
7
7                    IF .ST
7                    THEN
7                        LEAVE BLK;
7
6                    END;
6
6                ! if here there was a hard failure on either the first or second
6                ! getbkt
6                !
(                RAB[RAB$L_STV] = .ST;
6                IRAB[IRB$V_RRV_ERR] = 1;
6                LEAVE INNER;
6
5                END;                                  ! of local S1
6                BEGIN
6
6                LOCAL
6                    PTR     : REF BBLOCK;
6
6                PTR = .TABLE;
6
6                ! Do all the rrv's in this bucket that we have accessed.  Scan
6                ! through the rest of the table, comparing vbn's if we find one that
6                ! is the same as this one, take care of it now
6                !
6
6                WHILE .PTR LSSU .EOT
6                DO
7                    BEGIN
7
7                    IF .PTR[TBL$L_OLD_VBN] EQLU .TABLE[TBL$L_OLD_VBN]
7                    THEN
7 INNERMOST :
8                        BEGIN
8
```

```
BUILTIN
    AP;

IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
THEN
    AP = .PTR[TBL$B_OLD_ID]
ELSE
    AP = .PTR[TBL$W_OLD_ID];

BEGIN

LOCAL
    ST;

ST = RM$FIND_BY_ID();

! If bad status returned (ex: could not find by RFA)
! or this is NOT an RRV, or it is a DELETED RRV,
! then indicate error and mark entry done.
!
IF NOT .ST
    OR
    NOT .REC_ADDR[IRC$V_RRV]
    OR (.REC_ADDR[IRC$V_RRV] AND .REC_ADDR[IRC$V_DELETED])
THEN
    BEGIN

    ! Indicates that this table entry has been taken
    ! care of.
    !
    IF .PTR NEQ .TABLE
    THEN
        PTR[TBL$L_OLD_VBN] = 0;

    ! If the current table entry indicates that the
    ! corresponding record had not been deleted within a
    ! Recovery Unit, then as there must be a RRV for it
    ! somewhere, this inability to find one represents an
    ! error. Make sure that an RVU error will get returned
    ! in this case so the user knows to expect that some
    ! RRV pointers in the file will be incorrect.
    !
    IF NOT .PTR[TBL$V_RU_DELETE]
    THEN
        BEGIN
        RAB[RAB$L_STV] = .ST;
        IRAB[IRB$V_RRV_ERR] = 1;
        END;

    LEAVE INNERMOST;
    END;

END;                            ! { end of block defining st }

IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
THEN
    BEGIN
```

```
                    REC_ADDR[IRC$B_RRV_ID] = .PTR[TBL$B_NEW_ID];
                    REC_ADDR[IRC$L_RRV_VBN] = RM$SELECT_VBN(.PTR[TBL$B_NEW_VBN]);
                    END
                ELSE
                    BEGIN
                    REC_ADDR[IRC$W_RRV_ID] = .PTR[TBL$W_NEW_ID];
                    REC_ADDR[IR3$L_RRV_VBN] = RM$SELECT_VBN(.PTR[TBL$B_NEW_VBN]);
                    END;

                PTR[TBL$L_OLD_VBN] = 0;
                END;                    ! { end of vbns match -- innermost }

            PTR = .PTR + .ENTRY_SIZE;
            END;                                        ! {end of while loop }

        END;                                            ! of local PTR

        ! if we're done w/ this vbn, release it, writing it out
        !
        BEGIN
        BDB[BDB$V_DRT] = 1;
        BEGIN

        LOCAL
            ST;

        IF NOT (ST = RM$RLSBKT(RLS$M_WRT_THRU))
        THEN
            BEGIN
            RAB[RAB$L_STV] = .ST;
            IRAB[IRB$V_RRV_ERR] = 1;
            LEAVE INNER

            END;

        END;        ! { end of block defining st for call to rlsbkt }
        END;
        END;                ! {end of table entry is valid -- inner }

    TABLE = .TABLE + .ENTRY_SIZE;
    END;                                                ! {end of while loop }

END;                                                    ! { end of block }

! Release the buffer we used as a work space can't use rm$rlsbkt since it
! makes too many checks & i've clobbered the buffer
!
BDB = .IRAB[IRB$L_NXTBDB];
IRAB[IRB$L_NXTBDB] = 0;
BDB[BDB$B_FLGS] = 0;
RM$RELEASE(0);
END;
```

```
                                 0C   BB 00000 RM$UPDATE_RRV_2::
                                                       PUSHR   #^M<R2,R3>                      ; 1038
                 5E              0C   C2 00002         SUBL2   #12, SP                         ;
                 50        3C    A9   D0 00005         MOVL    60(IRAB), R0                    ; 1103
                 53        18    A0   D0 00009         MOVL    24(R0), TABLE                   ;
                 50              63   3C 0000D         MOVZWL  (TABLE), R0                     ; 1104
            04   AE            8340   9E 00010         MOVAB   (TABLE)+[R0], EOT               ;
                                 53   D6 00015         INCL    TABLE                           ; 1105
            03             00B7  CA   91 00017         CMPB    183(IFAB), #3                   ; 1107
                                 05   1E 0001C         BGEQU   1$                              ;
                 6E              07   D0 0001E         MOVL    #7, ENTRY_SIZE                  ; 1109
                                 03   11 00021         BRB     2$                              ;
                 6E              0A   D0 00023 1$:     MOVL    #10, ENTRY_SIZE                 ; 1111
            04   AE              53   D1 00026 2$:     CMPL    TABLE, EOT                      ; 1116
                                 03   1F 0002A         BLSSU   3$                              ;
                               00FD   31 0002C         BRW     20$                             ;
            08   AE        01    A3   D0 0002F 3$:     MOVL    1(TABLE), 8(SP)                 ; 1124
                                 03   12 00034         BNEQ    4$                              ;
                               00ED   31 00036         BRW     19$                             ;
                 52        17    A7   9A 00039 4$:     MOVZBL  23(IDX_DFN), SIZE               ; 1138
       52        52              09   78 0003D         ASHL    #9, SIZE, SIZE                  ;
            40   A9              01   90 00041         MOVB    #1, 64(IRAB)                    ; 1139
       04   06   A9              05   E1 00045         BBC     #5, 6(IRAB), 5$                 ; 1145
            40   A9              02   88 0004A         BISB2   #2, 64(IRAB)                    ; 1147
                                 52   DD 0004E 5$:     PUSHL   SIZE                            ; 1149
                           0C    AE   DD 00050         PUSHL   12(SP)                          ;
                              0000G   30 00053         BSBW    RM$GETBKT                       ;
                 5E              08   C0 00056         ADDL2   #8, SP                          ;
                 51              50   D0 00059         MOVL    R0, ST                          ;
                 36              51   E8 0005C         BLBS    ST, 7$                          ; 1151
          82AA   8F              51   B1 0005F         CMPW    ST, #33450                      ; 1155
                                 28   12 00064         BNEQ    6$                              ;
            06   A9              20   8A 00066         BICB2   #32, 6(IRAB)                    ; 1163
                 54      0084    C9   D0 0006A         MOVL    132(IRAB), BDB                  ; 1164
                         0084    C9   D4 0006F         CLRL    132(IRAB)                       ; 1165
                                 7E   D4 00073         CLRL    -(SP)                           ; 1166
                              0000G   30 00075         BSBW    RM$RLSBKT                       ;
            40   A9              01   90 00078         MOVB    #1, 64(IRAB)                    ; 1170
                 6E              52   D0 0007C         MOVL    SIZE, (SP)                      ; 1171
                           0C    AE   DD 0007F         PUSHL   12(SP)                          ;
                              0000G   30 00082         BSBW    RM$GETBKT                       ;
                 5E              08   C0 00085         ADDL2   #8, SP                          ;
                 51              50   D0 00088         MOVL    R0, ST                          ;
                 07              51   E8 0008B         BLBS    ST, 7$                          ; 1173
            0C   A8              51   D0 0008E 6$:     MOVL    ST, 12(RAB)                     ; 1182
                               008D   31 00092         BRW     18$                             ; 1183
                 52              53   D0 00095 7$:     MOVL    TABLE, PTR                      ; 1192
            04   AE              52   D1 00098 8$:     CMPL    PTR, EOT                        ; 1199
                                 71   1E 0009C         BGEQU   17$                             ;
            08   AE        01    A2   D1 0009E         CMPL    1(PTR), 8(SP)                   ; 1203
                                 65   12 000A3         BNEQ    16$                             ;
            03           00B7    CA   91 000A5         CMPB    183(IFAB), #3                   ; 1211
                                 06   1E 000AA         BGEQU   9$                              ;
                 5C        06    A2   9A 000AC         MOVZBL  6(PTR), AP                      ; 1213
                                 04   11 000B0         BRB     10$                             ;
                 5C        07    A2   3C 000B2 9$:     MOVZWL  7(PTR), AP                      ; 1215
                              0000G   30 000B6 10$:    BSBW    RM$FIND_BY_ID                   ; 1222
```

```
                    08          50  E9 000B9          BLBC    ST, 11$                          ; 1228
          04        66          03  E1 000BC          BBC     #3, (REC_ADDR), 11$              ; 1230
          16        66          02  E1 000C0          BBC     #2, (REC_ADDR), 13$              ; 1231
                    53          52  D1 000C4 11$:     CMPL    PTR, TABLE                       ; 1238
                                03  13 000C7          BEQL    12$                              ;
                          01    A2  D4 000C9          CLRL    1(PTR)                           ; 1240
                    3A    09    A2  E8 000CC 12$:     BLBS    9(PTR), 16$                      ; 1250
               0C   A8          50  D0 000D0          MOVL    ST, 12(RAB)                      ; 1253
               06   A9          04  88 000D4          BISB2   #4, 6(IRAB)                      ; 1254
                                30  11 000D8          BRB     16$                              ; 1257
                    03  00B7    CA  91 000DA 13$:     CMPB    183(IFAB), #3                    ; 1262
                                14  1E 000DF          BGEQU   14$                              ;
               02   A6    05    A2  90 000E1          MOVB    5(PTR), 2(REC_ADDR)              ; 1265
                    7E          62  9A 000E6          MOVZBL  (PTR), -(SP)                     ; 1266
                              FBDB  30 000E9          BSBW    RM$SELECT_VBN                    ;
                    5E          04  C0 000EC          ADDL2   #4, SP                           ;
               03   A6          50  D0 000EF          MOVL    R0, 3(REC_ADDR)                  ;
                                12  11 000F3          BRB     15$                              ; 1262
               03   A6    05    A2  B0 000F5 14$:     MOVW    5(PTR), 3(REC_ADDR)              ; 1270
                    7E          62  9A 000FA          MOVZBL  (PTR), -(SP)                     ; 1271
                              FBC7  30 000FD          BSBW    RM$SELECT_VBN                    ;
                    5E          04  C0 00100          ADDL2   #4, SP                           ;
               05   A6          50  D0 00103          MOVL    R0, 5(REC_ADDR)                  ;
                          01    A2  D4 00107 15$:     CLRL    1(PTR)                           ; 1274
                    52          6E  C0 0010A 16$:     ADDL2   ENTRY_SIZE, PTR                  ; 1277
                                89  11 0010D          BRB     8$                               ; 1199
               0A   A4          02  88 0010F 17$:     BISB2   #2, 10(BDB)                      ; 1285
                                02  DD 00113          PUSHL   #2                               ; 1291
                             0000G  30 00115          BSBW    RM$RLSBKT                        ;
                    5E          04  C0 00118          ADDL2   #4, SP                           ;
                    08          50  E8 0011B          BLBS    ST, 19$                          ;
               0C   A8          50  D0 0011E          MOVL    ST, 12(RAB)                      ; 1294
               06   A9          04  88 00122 18$:     BISB2   #4, 6(IRAB)                      ; 1295
                    53          6E  C0 00126 19$:     ADDL2   ENTRY_SIZE, TABLE                ; 1304
                              FEFA  31 00129          BRW     2$                               ; 1116
                    54    3C    A9  D0 0012C 20$:     MOVL    60(IRAB), BDB                    ; 1312
                          3C    A9  D4 00130          CLRL    60(IRAB)                         ; 1313
                          0A    A4  94 00133          CLRB    10(BDB)                          ; 1314
                                53  D4 00136          CLRL    [illegible]                      ; 1315
                      00000000G 00  16 00138          JSB     RM$RELEASE                       ;
                    5E          0C  C0 0013E          ADDL2   #12, SP                          ; 1316
                                0C  BA 00141          POPR    #^M<R2,R3>                       ;
                                05  00143             RSB                                      ;

; Routine Size:  324 bytes,    Routine Base:  RM$RMS3 + 0361


; 1258          1317  1
; 1259          1318  1 END
; 1260          1319  1
; 1261          1320  0 ELUDOM
;                                PSECT SUMMARY
;
;      Name                      Bytes                          Attributes
;
;  RM$RMS3                          1189  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  GBL,  REL,  CON,  PIC,ALIGN(2)




;                           Library Statistics
;
;                                            -------- Symbols --------       Pages       Processing
;      File                                  Total    Loaded   Percent       Mapped      Time
;
;  _$255$DUA28:[RMS.OBJ]RMS.L32;1             3109        78         2        154         00:00.4





;                                 COMMAND QUALIFIERS

;      BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3RRV/OBJ=OBJ$:RM3RRV MSRC$:RM3RRV/UPDATE=(ENH$:RM3RRV)

; Size:          1189 code + 0 data bytes
; Run Time:         00:29.7
; Elapsed Time:     00:57.4
; Lines/CPU Min:     2669
; Lexemes/CPU-Min: 17387
; Memory Used:  302 pages
; Compilation Complete
```

0327 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RM3PROBE
LIS

RM3SIDXSP
LIS

RM3PUTERR
LIS

RM3PUTUPD
LIS

RM3SPLUDR
LIS

RM3RRV
LIS

RM3ROOT
LIS

RM3PUT
LIS