```
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**RM3PUTUPD                                    G 5                                                                        RM
                                                                                                                                   VO

          RRRRRRRR    MM      MM    333333    PPPPPPPP   UU      UU  TTTTTTTTTT  UU      UU  PPPPPPPP   DDDDDDDD
          RRRRRRRR    MM      MM    333333    PPPPPPPP   UU      UU  TTTTTTTTTT  UU      UU  PPPPPPPP   DDDDDDDD
          RR      RR  MMMM  MMMM  33      33  PP      PP UU      UU      TT      UU      UU  PP      PP DD      DD
          RR      RR  MMMM  MMMM  33      33  PP      PP UU      UU      TT      UU      UU  PP      PP DD      DD
          RR      RR  MM  MM  MM          33  PP      PP UU      UU      TT      UU      UU  PP      PP DD      DD
          RR      RR  MM  MM  MM          33  PP      PP UU      UU      TT      UU      UU  PP      PP DD      DD
          RRRRRRRR    MM      MM        33    PPPPPPPP   UU      UU      TT      UU      UU  PPPPPPPP   DD      DD
          RRRRRRRR    MM      MM        33    PPPPPPPP   UU      UU      TT      UU      UU  PPPPPPPP   DD      DD
          RR  RR      MM      MM          33  PP         UU      UU      TT      UU      UU  PP         DD      DD
          RR  RR      MM      MM          33  PP         UU      UU      TT      UU      UU  PP         DD      DD
          RR    RR    MM      MM  33      33  PP         UU      UU      TT      UU      UU  PP         DD      DD
          RR    RR    MM      MM  33      33  PP         UU      UU      TT      UU      UU  PP         DD      DD    ....
          RR      RR  MM      MM    333333    PP         UUUUUUUUUU      TT      UUUUUUUUUU  PP         DDDDDDDD      ....
          RR      RR  MM      MM    333333    PP         UUUUUUUUUU      TT      UUUUUUUUUU  PP         DDDDDDDD      ....
                                                                                                                      ....

          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 RM3PUTUPD (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 ! FACILITY:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
1 !
1 ! ABSTRACT:
1 !                Contains routines common to both put and update operations
1 !
1 !
1 ! ENVIRONMENT:
1 !
1 !               VAX/VMS OPERATING SYSTEM
1 !
1 !--
1
1 !
1 ! AUTHOR:        Christian Saether          CREATION DATE:            4-OCT-78  13:45
1 !
1 ! Modified by:
1 !
1 !       V03-020 MCN0017         Maria del C. Nasr       04-Apr-1983
1 !               Change linkage of RM$NULLKEY to RL$JSB.
1 !
1 !       V03-019 TMK0012         Todd M. Katz            26-Mar-1983
1 !               Change the linkage for RM$RU_JOURNAL3 from RL$RABREG_467 to
1 !               RL$RABREG_67.
1 !
1 !       V03-018 MCN0016         Maria del C. Nasr       24-Mar-1983
1 !               More linkages reorganization.
1 !
```

```
1 !      V03-017 TMK0011         Todd M. Katz            16-Mar-1983
1 !              Change the linkage for RM$RU_JOURNAL3 from RL$RABREG_67 to
1 !              RL$RABREG_467.
1 !
1 !      V03-016 TMK0010         Todd M. Katz            16-Mar-1983
1 !              Change the symbol RMSR$_PUT to RJR$_PUT.
1 !
1 !      V03-015 MCN0015         Maria del C. Nasr       28-Feb-1983
1 !              Reorganize linkages
1 !
1 !      V03-014 TMK0009         Todd M. Katz            14-Jan-1983
1 !              Add support for Recovery Unit Journalling and RU ROLLBACK
1 !              Recovery of ISAM files. One modification will have to be made
1 !              to the routine RM$PUT_UPD_SPL. If the current operation causing
1 !              the primary data bucket split is a $PUT, then RU Journal the
1 !              operation before releasing the original bucket provided the file
1 !              is RU Journallable and the process is within a Recovery Unit.
1 !
1 !      V03-013 TMK0008         Todd M. Katz            29-Sep-1982
1 !              The $UPDATE RMS service has been completely re-written such
1 !              that it is no longer necessary to keep a copy of the old
1 !              version of the current primary data record (the record being
1 !              updated) in a scratch buffer or at the back end of the primary
1 !              data bucket when there are old secondary keys to be deleted.
1 !              The old record now has its own record buffer to reside in.
1 !              Therefore, it is no longer necessary to move the old record
1 !              out of the back end of the old (or leftmost) primary data bucket
1 !              into the back end of the scratch buffer used to build the RRV
1 !              table before the old primary data bucket is released.
1 !
1 !      V03-012 TMK0007         Todd M. Katz            09-Sep-1982
1 !              The field IRB$B_SRCHFLAGS has been changed to a word. Fix all
1 !              references to it.
1 !
1 !              The only time it in necessary to verify a packed decimal key
1 !              is when the key type is packed decimal. It is never necessary
1 !              to perform this verification when the key consists of more
1 !              than one segment and the file is a prologue 3 file. Also, the
1 !              packed decimal verification routine no longer requires
1 !              parameters.
1 !
1 !      V03-011 KBT0230         Keith B. Thompson       23-Aug-1982
1 !              Reorganize psects
1 !
1 !      V03-010 TMK0006         Todd M. Katz            19-Jul-1982
1 !              The name of RM$CLEAN_SIDR has been changed to RM$PUTUPD_ERROR.
1 !              There is also no longer any need to place the address of the
1 !              user record buffer (RAB$L_RBF) in the global register REC_ADDR
1 !              before calling the routine RM$PUTUPD_ERROR since that is done
1 !              by the routine itself.
1 !
1 !      V03-009 TMK0005         Todd M. Katz            02-Jul-1982
1 !              Implement the RMS cluster solution for next record positioning.
1 !              Since the NRP list has been eliminated and the next record
1 !              positioning context kept locally in the IRAB, there is no
1 !              need to update the NRP list when inserting (or re-inserting)
1 !              a record, and the RFA of the new/changed primary data record
```

```
!                maybe retrieved from the IRAB.
!
!       V03-008 KBT0071         Keith B. Thompson       28-Jun-1982
!               Modify the update buffer processing
!
!       V03-007 MCN0014         Maria del C. Nasr       11-Jun-1982
!               Eliminate overhead at end of data bucket that was to be
!               used for duplicate continuation bucket processing.
!
!       V03-006 KBT0058         Keith B. Thompson       9-Jun-1982
!               Change rm$ins_all_sidr to use rm$get_next_key and remove
!               all ref. to the significance count
!
!       V03-005 TMK0004         Todd M. Katz            07-Jun-1982
!               Whenever we have decided to treat multi-bucket splits involving
!               continuation buckets as two-bucket splits, the rightmost bucket
!               is a continuation bucket, and an index update is required, set
!               IRB$L_VBN_RIGHT to zero. This is because whatever index update
!               is done it will not involve the rightmost new bucket.
!
!       V03-004 TMK0003         Todd M. Katz            26-May-1982
!               Fix an assembly error that broke the build.
!
!       V03-003 TMK0002         Todd M. Katz            19-May-1982
!               There are several multi-bucket split cases involving
!               continuation but not empty buckets where the decision is made
!               to perform the index update as if a two-bucket data level
!               split had taken place. Towards this end, the IRB$V_BIG_SPLIT
!               flag is cleared. However, whenever an index update takes
!               place, there exists the possibility of an index bucket split,
!               and the two-pass index bucket split code for those cases not
!               involving empty buckets assumes that if some value is in
!               IRB$L_VBN_MID then a multi-bucket split must have taken place
!               on the level below. This is regardless of whether the
!               IRB$V_BIG_SPLIT flag is or isn't set. To resolve this conflict
!               we must also clear IRB$L_VBN_MID in addition to IRB$V_BIG_SPLIT
!               whenever we have decided to handle the index update as if a
!               two-bucket split has taken place, and whenever empty buckets
!               are not involved. This fix will be included as a patch on the
!               V3.1 update floppy.
!
!       V03-002 RAS0085         Ron Schaefer            8-Apr-1982
!               Correct DJD001 so that packed decimal keys get correctly
!               probed and checked.
!
!       V03-001 TMK0001         Todd M. Katz            24-March-1982
!               Change all references to the keybuffers, so that they use
!               the macro KEYBUF_ADDR.
!
!       V02-028 DJD0001         Darrell Duffy           1-March-1982
!               Fix probing of packed decimal key strings and
!               RM$PUT_UPD_CHKS.
!
!       V02-027 KBT0006         K B Thompson            15-Feb-1982
!               Add code to handle reclaimed bucket next-record-IDs and add
!               subtitles
!
```

```
1 !       V02-026 CDS0001         C Saether               30-Aug-1981
1 !               Update irb$l_curb-b after release with keep lock, as
1 !               it now represents the lock blb.
1 !
1 !       V02-025 MCN0013         Maria del C. Nasr       19-Aug-1981
1 !               Store the key in keybuffer3 to keybuffer5 when doing a big
1 !               split.
1 !
1 !       V02-024 MCN0012         Maria del C. Nasr       03-Aug-1981
1 !               Change calling sequence to RM$PCKDEC_CHECK to support
1 !               key type conversion.
1 !
1 !       V02-023 MCN0011         Maria del C. Nasr       27-May-1981
1 !               Modify RM$PUT_UPD_SPL to support bucket splitting in
1 !               $PUT for prologue 3 files.
1 !
1 !       V02-022 MCN0010         Maria del C. Nasr       15-May-1981
1 !               Change RM$PUT_UPD_CHCKS to include new overhead when
1 !               determining if prologue 3 records fit in bucket.
1 !
1 !       V02-021 MCN0006         Maria del C. Nasr       13-Mar-1981
1 !               Increase size of record identifier to a word in NRP.
1 !
1 !       V02-020 PSK0002         P S Knibbe      27-Aug-1980     17:00
1 !               If insert fails because secondary key is an invalid
1 !               packed decimal string, back out what has already been
1 !               inserted.
1 !
1 !       V02-019 REFORMAT        C Saether       01-Aug-1980     22:05
1 !
1 !
1 ! REVISION HISTORY:
1 !
1 !   Wendy Koenig,       24-OCT-78  14:02
1 !   X0002 - MAKE CHANGES CAUSED BY SHARING CONVENTIONS
1 !
1 !   Christian Saether,  24-OCT-78  17:41
1 !   X0003 - move original record (update only) out of CURBDB before calling
1 ! UPDATE_RRV
1 !
1 !   Christian Saether,  3-NOV-78  8:19
1 !   X0004 - fix incorrect use of BDB$W_SIZE to BDB$W_NUMB
1 !
1 !   Wendy Koenig,       6-NOV-78  13:32
1 !   X0005 - CHANGE OK_RRV (CONDITIONAL SUCCESS) TO RVU ( ERROR)
1 !
1 !   Christian Saether,  6-NOV-78  15:51
1 !   X0006 - fix bug masing BIG_SPLIT and EMPTY_BKT in SPL_BITS
1 !
1 !   Christian Saether,  9-NOV-78  10:40
1 !   X0007 - back to MAKE_INDEX on INS_ALL_SIDR
1 !
1 !   Wendy Koenig,       15-DEC-78  12:59
1 !   X0008 - FIX UP EMPTY BKT CASES FOR FOOL_UPD
1 !
1 !   Christian Saether,  14-JAN-79  17:03
1 !   X0009 - do not set lockabove when root is at level 1 - causes deadlock
```

```
1 .
1 .    Wendy Koenig,        15-JAN-79  12:09
1 !    X0010 - DELETE LINE OF CODE THAT SETS VALID AND DIRTY BITS FOR NO REASON
1 .
1 .    Christian Saether,  15-JAN-79  17:54
1 !    X0011 - take LOCKABOVE decision out of INS_ALL_SIDR
1 !
1 !    Wendy Koenig,        25-JAN-79  12:20
1 !    X0012 - FOOL UPDATE WHEN EMPTY BKT AND NOT BIG SPLIT
1 !
1 .    Wendy Koenig,        26-JAN-79  9:14
1 !    X0013 - GET RID OF SETTING VALID
1 !
1 .    Wendy Koenig,        9-FEB-79  14:30
1 .    X0014 - CLEAR UNLOCK_RP BIT WHEN UNLOCKING RECORD
1 .            RETURN ERROR IF RRV BIT IS SET FROM UPDATE_RRV_2
1 !                 SO WE DON'T HAVE TO CHECK BIT LATER
1 !
1 !    Christian Saether,  11-FEB-79  19:43
1 !    X0015 - take record unlocking logic out of put_upd_fin (moved to rm3face)
1 !
1 !    Christian Saether,  1-july-79  11:20
1 !    X0016 - clear irb$v_dup in rm$put_upd_chks
1 !
1 !    Paulina Knibbe,      19-Dec-79  11:30
1 !    X0017 - check that packed decimal keys are legal
1 !
1 !    Christian Saether,  1-Jan-80  22:35
1 !    0018 - do bucket sort to recover id's if splitting due to lack of id's
1 !
1 !*****
1
1 LIBRARY 'RMSLIB:RMS';
1
1 REQUIRE 'RMSSRC:RMSIDXDEF';
1
1 ! define default psects for code
1 !
1 PSECT
1     CODE = RM$RMS3(PSECT_ATTR),
1     PLIT = RM$RMS3(PSECT_ATTR);
1
1 ! Linkages
1 !
1 LINKAGE
1     L_ERROR_LINK2,
1     L_JSB,
1     L_LINK_7_10_11,
1     L_PRESERVE1,
1     L_RABREG,
1     L_RABREG_4567,
1     L_RABREG_457,
1     L_RABREG_567,
1     L_RABREG_67,
1     L_RABREG_7;
1
1 ! External Routines
```

```
1 !
1 EXTERNAL ROUTINE
1     RM$ALLOC_BKT           : RL$RABREG_7,
1     RM$BKT_SORT            : RL$RABREG_7,
1     RM$BKT_SPL             : RL$RABREG_67,
1     RM$FOOLED_YUH          : RL$RABREG_7,
1     RM$GETBKT              : RL$RABREG_457,
1     RM$GET_NEXT_KEY        : RL$LINK_7_10_11,
1     RM$INSERT_REC          : RL$RABREG_4567,
1     RM$INSS_OR_IDX         : RL$RABREG_567,
1     RM$KEY_DESC            : RL$RABREG_7,
1     RM$MAKE_INDEX          : RL$RABREG_7,
1     RM$MOVE                : RL$PRESERVE1,
1     RM$NOREAD_LONG         : RL$JSB,
1     RM$NULLKEY             : RL$JSB,
1     RM$PCKDEC_CHECK        : RL$RABREG_7,
1     RM$PUTUPD_ERROR        : RL$ERROR_LINK2,
1     RM$RECORD_KEY          : RL$PRESERVE1,
1     RM$RLSBKT              : RL$PRESERVE1,
1     RM$RU_JOURNAL3         : RL$RABREG_67,
1     RM$SPLIT_UDR           : RL$RABREG_4567 NOVALUE,
1     RM$SPLIT_UDR_3         : RL$RABREG_4567 NOVALUE,
1     RM$UPDATE_RRV          : RL$RABREG_67,
1     RM$UPDATE_RRV_2        : RL$RABREG_4567;
1
```

```
1 %SBTTL  'RM$PUT_UPD_SPL'
1 GLOBAL ROUTINE RM$PUT_UPD_SPL(RECSZ) : RL$RABREG_4567 =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Handles the UDR split code for $PUT and $UPDATE
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       BSBW RM$PUT_UPD()
1 !
1 ! INPUT PARAMETERS:
1 !
1 !       RECSZ - record size of record to be inserted
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       REC_ADDR -- position for insert
1 !       IDX_DFN -- index descriptor for primary key
1 !       RAB, IRAB, IFAB, IMPURE
1 !       in RAB -- RSZ, RBF
1 !       keybuffers
1 !       in IRAB -- UNLOCK_RP UPDATE bits, CURBDB describing bucket to insert in
1 !               -- if non-zero, LOCK BDB describes level above in case of idx upd
1 !               -- RSZ and RBF
1 !       in IFAB -- NORECLK RUP bits
1 !       BKT$B_NXTRECID = 0 in original bucket to signal that split was
1 !       due to lack of id's
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       RBF, RSZ, DUP, any hard i/o errors
1 !       RVU -- some RRV(s) were not updated correctly
1 !       OK_DUP (dups seen)
1 !
1 ! SIDE EFFECTS:
1 !
1 !       if success of any type, the record has been put
1 !       if success then index has been updated
1 !       in any case, if UNLOCK_RP is set, the record is unlocked otherwise locked
1 !       note that it is locked throughout index and sidr updates
1 !       put has to unlock the record if there was an error, update is cool
1 !       no buckets are locked on exit in any case
1 !       IRAB [ UPDATE ] flag is set if index updating is required
1 !       If the current operation is a $PUT taking place on a RU Journallable
1 !           file within a Recovery Unit, then RU Journal the operation before
1 !             releasing the original bucket.
1 !
1 !--
```

```
1
2       BEGIN
2
2       EXTERNAL REGISTER
2           COMMON_IO_STR,
2           COMMON_RAB_STR,
2           R_IDX_DFN_STR,
2           R_REC_ADDR_STR;
2
2       LABEL
2           FOOL_UPD;
2
2       ! initialize all the fie'ds we're going to use
2       !
2       IRAB[IRB$W_SPLIT] = 0;
2       IRAB[IRB$W_SPLIT_1] = 0;
2       IRAB[IRB$W_SPLIT_2] = 0;
2       IRAB[IRB$L_RFA_VBN] = 0;
2       IRAB[IRB$L_VBN_LEFT] = 0;
2       IRAB[IRB$L_VBN_RIGHT] = 0;
2       IRAB[IRB$L_VBN_MID] = 0;
2
2       ! calculate the kind of split that's neccessary
2       !
2       IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
2       THEN
2           RM$SPLIT_UDR()
2       ELSE
2           RM$SPLIT_UDR_3(.RECSZ);
2       IRAB[IRB$L_VBN_LEFT] = .BDB[BDB$L_VBN];
2       IRAB[IRB$V_BKT_NO] = .IRAB[IRB$V_NEW_BKTS];
2
2       ! Store the primary key in keybuffer5 when doing a big split, so that
2       ! RM$BKT_SPL can save the significant count in keybuffer3 safely?
2       !
2       IF .IRAB[IRB$V_NEW_BKTS] GTRU 1
2       THEN
2           RM$MOVE ( .IDX_DFN[IDX$B_KEYSZ], KEYBUF_ADDR(3), KEYBUF_ADDR(5));
2
2       ! for each new bucket that's needed, allocate it fill in the data and write
2       ! it out
2       !
2       WHILE .IRAB[IRB$V_BKT_NO] NEQ 0
2       DO
3           BEGIN
3
3           ! allocate and format new bucket setting up nxtbdb to describe it
3           ! link to previous new bucket
3           !
3           RETURN_ON_ERROR (RM$ALLOC_BKT());
3
3           ! set up bdb and bkt_addr
3           !
3           BDB = .IRAB [ IRB$L_NXTBDB ];
3           BKT_ADDR = .BDB [ BDB$L_ADDR ];
3
3           ! Store vbn and next record ID in appropriate place for safe-keeping
```

```
3   ! the vbn's are stored for future use, as such:
3   !       the original bucket is always in vbn_left
3   !       the rightmost bucket is always in vbn_right
3   !       if there are 2 new buckets, the middle one is in vbn_mid
3   !       if there are 3 new buckets, the left-middle one is in vbn_mid
3   !               and the right-middle one is in rfa_vbn
3   ! the way split_udr is set up now, if it is a 3 or 4 bucket split
3   ! the new record is always in vbn_mid... but that of course is subject
3   ! to change
3   !
3   ! For each bucket save the VBN and the next record ID.  NOTE: The next
3   ! record ID for prologue 1,2 files id always 1
3   !
3   IF .IRAB [ IRB$L_VBN_RIGHT ] EQL 0
3   THEN
4       BEGIN
4
4       ! Save the VBN
4       !
4       IRAB [ IRB$L_VBN_RIGHT ] = .BDB [ BDB$L_VBN ];
4
4       ! Save the next record ID
4       !
4       IF .IFAB [ IFB$B_PLG_VER ] LSSU PLG$C_VER_3
4       THEN
4           IRAB [ IRB$W_NID_RIGHT ] = 1
4       ELSE
4           IRAB [ IRB$W_NID_RIGHT ] = .BBLOCK [ .BDB [ BDB$L_ADDR ],BKT$W_NXTRECID ]
4
4       END
3   ELSE
4       BEGIN
4
4       ! If the MID is taken this must be the RFA bucket
4       !
4       IF .IRAB [ IRB$L_VBN_MID ] NEQ 0
4       THEN
5           BEGIN
5
5           ! The old MID bucket becomes the RFA bucket
5           !
5           IRAB [ IRB$L_RFA_VBN ] = .IRAB [ IRB$L_VBN_MID ];
5           IRAB [ IRB$W_RFA_NID ] = .IRAB [ IRB$W_NID_MID ]
5
4           END;
4
4       ! Save the MID VBN
4       !
4       IRAB [ IRB$L_VBN_MID ] = .BDB [ BDB$L_VBN ];
4
4       ! Save the next rec rd ID
4       !
4       IF .IFAB [ IFB$B_PLG_VER ] LSSU PLG$C_VER_3
4       THEN
4           IRAB [ IRB$W_NID_MID ] = 1
4       ELSE
4           IRAB [ IRB$W_NID_MID ] = .BBLOCK [ .BDB [ BDB$L_ADDR ],BKT$W_NXTRECID ]
```

```

            END;

        ! Check to be sure the user hasn't been toying w/ the record buffer
        ! in less than prologue 3 files
        !
        IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
        THEN
            RETURN_ON_ERROR (RM$FOOLED_YUH());

        ! write out the data into the new bucket
        !
        RM$BKT_SPL(.RECSZ);                             ! no errors possible

        IRAB[IRB$L_NXTBDB] = 0;

        RETURN_ON_ERROR (RM$RLSBKT(RLS$M_WRT_THRU));

        END;    ! { end of while there are still more buckets to take care of }


    ! update rrv's, building table for future updates.
    ! set up the free buffer to be a table in which to build rrv's.
    ! let nxtbdb point to it and curbdb still points to the original bucket.
    !
    IRAB[IRB$B_CACHEFLGS] = CSH$M_NOREAD;
    RM$GETBKT(0, .BDB[BDB$W_NUMB]);
    IRAB[IRB$L_NXTBDB] = .BDB;

    RM$UPDATE_RRV();

    ! Set up bdb and bkt_addr to correspond to original bucket. Do bucket sort
    ! to recover id's if nxtrecid = 0 before potentially inserting new record
    ! into original bucket.  Note that we may be doing bucket sort in a put
    ! situations where nxtrecid happened to be zero on normal lack of space
    ! split, but it seems harmless enough and probably saves the need to do one
    ! later and zeroing nxtrecid to signal split due to lack of id's in the
    ! first place saves code in rm$split_udr.  (Only for prologue 1 and 2 files).
    !
    BDB = .IRAB[IRB$L_CURBDB];
    BKT_ADDR = .BDB[BDB$L_ADDR];

    IF (NOT .IRAB[IRB$V_UPDATE]
        AND
        .BKT_ADDR[BKT$B_NXTRECID] EQL 0)
        AND .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
    THEN
        RM$BKT_SORT(.BKT_ADDR);

    ! if the new record belongs in the original bucket, put it there
    !

    IF .IRAB[IRB$W_POS_INS] LSSU .IRAB[IRB$W_SPLIT]
        OR
        (.IRAB[IRB$W_POS_INS] EQLU .IRAB[IRB$W_SPLIT]
        AND
        .IRAB[IRB$V_REC_W_LO])
```

```
  2     THEN
  3        BEGIN
  3
  3        ! check again to make sure the user hasn't changed the record buffer
  3        !
  3
  3        IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
  3        THEN
  3           RETURN_ON_ERROR (RM$FOOLED_YUH());
  3
  3        REC_ADDR = .IRAB[IRB$W_POS_INS] + .BKT_ADDR;
  3
  3        RM$INSERT_REC(.RECSZ);                     ! always returns succesful
  2        END;            ! { end of record needs to go into original bucket }
  2
  2     ! If this operation is a $PUT taking place on a RU Journallable file within
  2     ! a Recovery Unit, then RU Journal the operation before releasing the
  2     ! original primary data bucket.
  2     !
  2     IF  .IFAB[IFB$V_RUP]
  2          AND
  2        NOT .IRAB[IRB$V_UPDATE]
  2     THEN
  3        BEGIN
  3        REC_ADDR = .IRAB[IRB$L_RBF];
P 3        RETURN_ON_ERROR (RM$RU_JOURNAL3 (RJR$_PUT,
P 3                                     .IRAB[IRB$L_PUTUP_VBN],
P 3                                     .IRAB[IRB$W_PUTUP_ID],
  3                                     .IRAB[IRB$W_RSZ]));
  2        END;
  2
  2     ! Release the original bucket, keeping a lock on it
  2     !
  2     BDB[BDB$V_DRT] = 1;
  2
  2     RETURN_ON_ERROR (RM$RLSBKT(RLS$M_WRT_THRU OR RLS$M_KEEP_LOCK));
  2     IRAB[IRB$L_CURBDB] = .BDB;
  2
  2     RM$UPDATE_RRV_2();
  2
  2     ! if RRV bit is set, there's been an error -- not much sense in going
  2     ! on, so to be consistent with other errors, let's clean up now
  2     !
  2
  2     IF .IRAB[IRB$V_RRV_ERR]
  2     THEN
  2        RETURN RMSERR(RVU);
  2
  2     ! if update needed then perform setup and flag update necessary. if update
  2     ! isn't needed, clear update flag since we still have curbdb locked
  2     ! and we have to know if we should release it
  2     !
  2  FOOL_UPD :
  3     BEGIN
  3     IRAB[IRB$V_UPDATE] = 1;
  3
  3     ! If it's a big split but both new buckets are continuation bkts, no index
```

```
;  597          0660  3          ! update is needed. However if the left-hand bucket is empty, RMS will have
;  598          0661  3          ! to swing the pointer to IRB$L_VBN_MID. As this effectively makes the
;  599          0662  3          ! split a (one or two pass) two-bucket empty bucket split case, zero
;  600          0663  3          ! IRB$L_VBN_RIGHT.
;  601          0664  3          !
;  602          0665  3          IF .IRAB[IRB$V_BIG_SPLIT]
;  603          0666  3              AND
;  604          0667  3              .IRAB[IRB$V_CONT_BKT]
;  605          0668  3              AND
;  606          0669  3              .IRAB[IRB$V_CONT_R]
;  607          0670  3          THEN
;  608          0671  4              BEGIN
;  609          0672  4
;  610          0673  4              IF .IRAB[IRB$V_EMPTY_BKT]
;  611          0674  4              THEN
;  612          0675  5                  BEGIN
;  613          0676  5                  IRAB[IRB$V_BIG_SPLIT] = 0;
;  614          0677  5                  IRAB[IRB$L_VBN_RIGHT] = 0;
;  615          0678  5                  END
;  616          0679  4              ELSE
;  617          0680  4                  IRAB[IRB$V_UPDATE] = 0;
;  618          0681  4
;  619          0682  4              LEAVE FOOL_UPD
;  620          0683  4
;  621          0684  3              END;
;  622          0685  3
;  623          0686  3          ! if it's not a big split and if the new bkt is a cont. bkt we don't have
;  624          0687  3          ! to upd. if it's not a big split and the original bkt is now empty,
;  625          0688  3          ! we have to update but we have to move vbn_mid into right and zero right.
;  626          0689  3          !
;  627          0690  3          IF NOT .IRAB[IRB$V_BIG_SPLIT]
;  628          0691  3          THEN
;  629          0692  4              BEGIN
;  630          0693  4
;  631          0694  4              IF .IRAB[IRB$V_CONT_BKT]
;  632          0695  4              THEN
;  633          0696  4                  IRAB[IRB$V_UPDATE] = 0;
;  634          0697  4
;  635          0698  4              IF .IRAB[IRB$V_EMPTY_BKT]
;  636          0699  4              THEN
;  637          0700  5                  BEGIN
;  638          0701  5                  IRAB[IRB$L_VBN_MID] = .IRAB[IRB$L_VBN_RIGHT];
;  639          0702  5                  IRAB[IRB$L_VBN_RIGHT] = 0;
;  640          0703  4                  END;
;  641          0704  4
;  642          0705  4              LEAVE FOOL_UPD
;  643          0706  4
;  644          0707  3              END;
;  645          0708  3
;  646          0709  3          ! at this point, if it's not a big split and the above conditions weren't
;  647          0710  3          ! met we need to update, but there is nothing else we want to change, so
;  648          0711  3          ! get out
;  649          0712  3          !
;  650          0713  3          IF NOT .IRAB[IRB$V_BIG_SPLIT]
;  651          0714  3          THEN
;  652          0715  3              LEAVE FOOL_UPD;
;  653          0716  3
```

```
3        ! Now, we know it's a big split case... see if we can reduce it to a 2-bkt
3        ! split if the original bkt is empty, and the rightmost is a cont bkt.
3        ! To do so, turn off the big split indication, and set IRB$L_VBN_RIGHT
3        ! to 0 (and just swing the ptr if possible).
3        !
3        IF .IRAB[IRB$V_EMPTY_BKT]
3            AND
3            .IRAB[IRB$V_CONT_R]
3        THEN
4            BEGIN
4            IRAB[IRB$V_BIG_SPLIT] = 0;
4            IRAB[IRB$L_VBN_RIGHT] = 0;
4            LEAVE FOOL_UPD;
3            END;
3
3        ! if the original bucket is empty, we've changed all we can
3        !
3        IF .IRAB[IRB$V_EMPTY_BKT]
3        THEN
3            LEAVE FOOL_UPD;
3
3        ! if the middle bucket is a continuation bucket, just make it a 2-bkt split
3        !
3        IF .IRAB[IRB$V_CONT_BKT]
3        THEN
3            IRAB[IRB$V_BIG_SPLIT] = 0;
3
3        ! if the right bucket is a continuation bucket, make it a 2-bkt split
3        ! and treat the middle bucket as the right bucket
3        !
3        IF .IRAB[IRB$V_CONT_R]
3        THEN
4            BEGIN
4            IRAB[IRB$L_VBN_RIGHT] = .IRAB[IRB$L_VBN_MID];
4            IRAB[IRB$V_BIG_SPLIT] = 0;
3            END;
3
3        ! If we have decided to treate the index update as if a two-bucket data
3        ! level split had taken place, then we must also clear VBN_MID. The
3        ! two-pass nonempty bucket index bucket split code assumes, regardless of
3        ! the status of IRB$V_BIG_SPLIT, that if this field has a value then the
3        ! two-pass index bucket split should be handled as if a multi-buck^t
3        ! split had taken place.
3        !
3        IF NOT .IRAB[IRB$V_BIG_SPLIT]
3        THEN
3            IRAB[IRB$L_VBN_MID] = 0;
3
2        END;                                            ! { end of fool_upd }
2
2        IRAB[IRB$B_SPL_BITS] = .IRAB[IRB$B_SPL_BITS] AND
2                            (IRB$M_BIG_SPLIT OR IRB$M_EMPTY_BKT);
2        RETURN RMSSUC(SUC);
2
1        END;
```

```
                                                            .TITLE  RM3PUTUPD                                    ;
                                                            .IDENT  \V04-000\                                    ;
                                                                                                                 ;
                                                            .EXTRN  RM$ALLOC_BKT, RM$BKT_SORT                    ;
                                                            .EXTRN  RM$BKT_SPL, RM$FOOLED_YUH                    ;
                                                            .EXTRN  RM$GETBKT, RM$GET_NEXT_KEY                   ;
                                                            .EXTRN  RM$INSERT_REC, RM$INSS_OR_IDX                ;
                                                            .EXTRN  RM$KEY_DESC, RM$MAKE_INDEX                   ;
                                                            .EXTRN  RM$MOVE, RM$NOREAD_LONG                      ;
                                                            .EXTRN  RM$NULLKEY, RM$PCKDEC_CHECK                  ;
                                                            .EXTRN  RM$PUTUPD_ERROR                              ;
                                                            .EXTRN  RM$RECORD_KEY, RM$RLSBKT                     ;
                                                            .EXTRN  RM$RU_JOURNAL3, RM$SPLIT_UDR                 ;
                                                            .EXTRN  RM$SPLIT_UDR_3, RM$UPDATE_RRV                ;
                                                            .EXTRN  RM$UPDATE_RRV_2                              ;
                                                                                                                 ;
                                                            .PSECT  RM$RMS3,NOWRT,  GBL,  PIC,2                  ;
                                                                                                                 ;
                                           52  DD 00000 RM$PUT_UPD_SPL::                                         ;
                                                            PUSHL   R2                                     ; 0376 ;
                                   4A      A9  D4 00002     CLRL    74(IRAB)                               ; 0446 ;
                                   4E      A9  B4 00005     CLRW    78(IRAB)                               ; 0448 ;
                                   70      A9  D4 00008     CLRL    112(IRAB)                              ; 0449 ;
                                 0088      C9  7C 0000B     CLRQ    136(IRAB)                              ; 0450 ;
                                 0090      C9  D4 0000F     CLRL    144(IRAB)                              ; 0452 ;
                          03     00B7      CA  91 00013     CMPB    183(IFAB), #3                          ; 0456 ;
                                           05  1E 00018     BGEQU   1$                                     ;      ;
                                        0000G  30 0001A     BSBW    RM$SPLIT_UDR                           ; 0458 ;
                                           09  11 0001D     BRB     2$                                     ;      ;
                                   08      AE  DD 0001F 1$: PUSHL   RECSZ                                  ; 0460 ;
                                        0000G  30 00022     BSBW    RM$SPLIT_UDR_3                         ;      ;
                          5E               04  C0 00025     ADDL2   #4, SP                                 ;      ;
                    0088  C9           1C  A4  D0 00028 2$: MOVL    28(BDB), 136(IRAB)                     ; 0461 ;
       50     44    A9    02               01  EF 0002E     EXTZV   #1, #2, 68(IRAB), R0                   ; 0462 ;
  44   A9           02    00               50  F0 00034     INSV    R0, #0, #2, 68(IRAB)                   ;      ;
       01     44    A9    02               01  ED 0003A     CMPZV   #1, #2, 68(IRAB), #1                   ; 0467 ;
                                           17  1B 00040     BLEQU   3$                                     ;      ;
                          50     00B4      CA  3C 00042     MOVZWL  180(IFAB), R0                          ; 0469 ;
                                   60 B940     DF 00047     PUSHAL  @96(IRAB)[R0]                          ;      ;
                                   60 B940     3F 0004B     PUSHAW  @96(IRAB)[R0]                          ;      ;
                          7E       20      A7  9A 0004F     MOVZBL  32(IDX_DFN), -(SP)                     ;      ;
                                        0000G  30 00053     BSBW    RM$MOVE                                ;      ;
                          5E               0C  C0 00056     ADDL2   #12, SP                                ;      ;
                          03       44      A9  93 00059 3$: BITB    68(IRAB), #3                           ; 0474 ;
                                           03  12 0005D     BNEQ    4$                                     ;      ;
                                         0089  31 0005F     BRW     12$                                    ;      ;
                                        0000G  30 00062 4$: BSBW    RM$ALLOC_BKT                           ; 0481 ;
                          7D               50  E9 00065     BLBC    STATUS, 11$                            ;      ;
                          54       3C      A9  D0 00068     MOVL    60(IRAB), BDB                          ; 0485 ;
                          50       18      A4  D0 0006C     MOVL    24(BDB), R0                            ; 0486 ;
                          55               50  D0 00070     MOVL    R0, BKT_ADDR                           ;      ;
                          52     00B7      CA  9E 00073     MOVAB   183(IFAB), R2                          ; 0512 ;
                                 008C      C9  D5 00078     TSTL    140(IRAB)                              ; 0502 ;
                                           1E  12 0007C     BNEQ    6$                                     ;      ;
                    008C  C9           1C  A4  D0 0007E     MOVL    28(BDB), 140(IRAB)                     ; 0508 ;
                                           51  D4 00084     CLRL    R1                                     ; 0512 ;
                          03               62  91 00086     CMPB    (R2), #3                               ;      
```

```
                                09  1E 00089          BGEQU   5$                              ;
                                51  D6 0008B          INCL    R1                              ;
                00A0    C9      01  B0 0008D          MOVW    #1, 160(IRAB)                   ; 0514
                                36  11 00092          BRB     9$                              ;
                00A0    C9  06  A0  B0 00094 5$:      MOVW    6(R0), 160(IRAB)                ; 0516
                                2E  11 0009A          BRB     9$                              ; 0512
                        51  0090 C9 D0 0009C 6$:      MOVL    144(IRAB), R1                   ; 0524
                                0B  13 000A1          BEQL    7$                              ;
                  70    A9      51  D0 000A3          MOVL    R1, 112(IRAB)                   ; 0530
                00A4    C9  00A2 C9 B0 000A7          MOVW    162(IRAB), 164(IRAB)            ; 0531
                0090    C9  1C  A4  D0 000AE 7$:      MOVL    28(BDB), 144(IRAB)              ; 0537
                                51  D4 000B4          CLRL    R1                              ; 0541
                        03      62  91 000B6          CMPB    (R2), #3                        ;
                                09  1E 000B9          BGEQU   8$                              ;
                                51  D6 000BB          INCL    R1                              ;
                00A2    C9      01  B0 000BD          MOVW    #1, 162(IRAB)                   ; 0543
                                06  11 000C2          BRB     9$                              ;
                00A2    C9  06  A0  B0 000C4 8$:      MOVW    6(R0), 162(IRAB)                ; 0545
                        06      51  E9 000CA 9$:      BLBC    R1, 10$                         ; 0552
                              0000G 30 000CD          BSBW    RM$FOOLED_YUH                   ; 0554
                        68      50  E9 000D0          BLBC    STATUS, 15$                     ;
                            08  AE  DD 000D3 10$:     PUSHL   RECSZ                           ; 0558
                              0000G 30 000D6          BSBW    RM$BKT_SPL                      ;
                            3C  A9  D4 000D9          CLRL    60(IRAB)                        ; 0560
                        6E      02  D0 000DC          MOVL    #2, (SP)                        ; 0562
                              0000G 30 000DF          BSBW    RM$RLSBKT                       ;
                        5E      04  C0 000E2          ADDL2   #4, SP                          ;
                        53      50  E9 000E5 11$:     BLBC    STATUS, 15$                     ;
                              FF6E  31 000E8          BRW     3$                              ;
                  40    A9      04  90 000EB 12$:     MOVB    #4, 64(IRAB)                    ; 0571
                        7E  14  A4  3C 000EF          MOVZWL  20(BDB), -(SP)                  ; 0572
                                7E  D4 000F3          CLRL    -(SP)                           ;
                              0000G 30 000F5          BSBW    RM$GETBKT                       ;
                        5E      08  C0 000F8          ADDL2   #8, SP                          ;
                  3C    A9      54  D0 000FB          MOVL    BDB, 60(IRAB)                   ; 0573
                              0000G 30 000FF          BSBW    RM$UPDATE_RRV                   ; 0575
                        54  20  A9  D0 00102          MOVL    32(IRAB), BDB                   ; 0585
                        55  18  A4  D0 00106          MOVL    24(BDB), BKT_ADDR               ; 0586
          14      06    A9      03  E0 0010A          BBS     #3, 6(IRAB), 13$                ; 0588
                            06  A5  95 0010F          TSTB    6(BKT_ADDR)                     ; 0590
                                0F  12 00112          BNEQ    13$                             ;
                        03  00B7 CA 91 00114          CMPB    183(IFAB), #3                   ; 0591
                                08  1E 00119          BGEQU   13$                             ;
                                55  DD 0011B          PUSHL   BKT_ADDR                        ; 0593
                              0000G 30 0011D          BSBW    RM$BKT_SORT                     ;
                        5E      04  C0 00120          ADDL2   #4, SP                          ;
                  4A    A9  48  A9  B1 00123 13$:     CMPW    72(IRAB), 74(IRAB)              ; 0598
                                07  1F 00128          BLSSU   14$                             ;
                                22  12 0012A          BNEQ    17$                             ; 0600
          1D      44    A9      03  E1 0012C          BBC     #3, 68(IRAB), 17$               ; 0602
                        03  00B7 CA 91 00131 14$:     CMPB    183(IFAB), #3                   ; 0609
                                06  1E 00136          BGEQU   16$                             ;
                              0000G 30 00138          BSBW    RM$FOOLED_YUH                   ; 0611
                        56      50  E9 0013B 15$:     BLBC    STATUS, 19$                     ;
                        56  48  A9  3C 0013E 16$:     MOVZWL  72(IRAB), REC_ADDR              ; 0613
                        56      55  C0 00142          ADDL2   BKT_ADDR, REC_ADDR              ;
                            08  AE  DD 00145          PUSHL   RECSZ                           ; 0615
```

```
                                    0000G 30 00148          BSBW    RM$INSERT_REC                           ;
                  5E                   04 C0 0014B          ADDL2   #4, SP                                  ;
    20      00A2  CA                   02 E1 0014E 17$:     BBC     #2, 162(IFAB), 18$                      ; 0622
    1B        06  A9                   03 E0 00154          BBS     #3, 6(IRAB), 18$                        ; 0624
                  56            58     A9 D0 00159          MOVL    88(IRAB), REC_ADDR                      ; 0627
                  7E            56     A9 3C 0015D          MOVZWL  86(IRAB), -(SP)                         ; 0631
                  7E          0080     C9 3C 00161          MOVZWL  128(IRAB), -(SP)                        ;
                                78     A9 DD 00166          PUSHL   120(IRAB)                               ;
                                       13 DD 00169          PUSHL   #19                                     ;
                                    0000G 30 0016B          BSBW    RM$RU_JOURNAL3                          ;
                  5E                   10 C0 0016E          ADDL2   #16, SP                                 ;
                  20                   50 E9 00171          BLBC    STATUS, 19$                             ;
              0A  A4                   02 88 00174 18$:     BISB2   #2, 10(BDB)                             ; 0636
                                       06 DD 00178          PUSHL   #6                                      ; 0638
                                    0000G 30 0017A          BSBW    RM$RLSBKT                               ;
                  5E                   04 C0 0017D          ADDL2   #4, SP                                  ;
                  11                   50 E9 00180          BLBC    STATUS, 19$                             ;
              20  A9                   54 D0 00183          MOVL    BDB, 32(IRAB)                           ; 0639
                                    0000G 30 00187          BSBW    RM$UPDATE_RRV_2                         ; 0641
    07        06  A9                   02 E1 0018A          BBC     #2, 6(IRAB), 20$                        ; 0647
                  50          868C     8F 3C 0018F          MOVZWL  #34444, R0                              ; 0649
                                       70 11 00194 19$:     BRB     32$                                     ;
              06  A9                   08 88 00196 20$:     BISB2   #8, 6(IRAB)                             ; 0657
                  50            44     A9 9E 0019A          MOVAB   68(IRAB), R0                            ; 0665
    16            60                   02 E1 0019E          BBC     #2, (R0), 22$                           ;
    0E            60                   04 E1 001A2          BBC     #4, (R0), 21$                           ; 0667
    0A            60                   05 E1 001A6          BBC     #5, (R0), 21$                           ; 0669
    27            60                   06 E0 001AA          BBS     #6, (R0), 25$                           ; 0673
              06  A9                   08 8A 001AE          BICB2   #8, 6(IRAB)                             ; 0680
                                       4B 11 001B2          BRB     31$                                     ; 0682
    15            60                   02 E0 001B4 21$:     BBS     #2, (R0), 24$                           ; 0690
    04            60                   04 E1 001B8 22$:     BBC     #4, (R0), 23$                           ; 0694
              06  A9                   08 8A 001BC          BICB2   #8, 6(IRAB)                             ; 0696
    3B            60                   06 E1 001C0 23$:     BBC     #6, (R0), 31$                           ; 0698
            0090  C9          008C     C9 D0 001C4          MOVL    140(IRAB), 144(IRAB)                    ; 0701
                                       0B 11 001CB          BRB     26$                                     ; 0702
    11            60                   06 E1 001CD 24$:     BBC     #6, (R0), 28$                           ; 0722
    09            60                   05 E1 001D1          BBC     #5, (R0), 27$                           ; 0724
                  60                   04 8A 001D5 25$:     BICB2   #4, (R0)                                ; 0727
                              008C     C9 D4 001D8 26$:     CLRL    140(IRAB)                               ; 0728
                                       21 11 001DC          BRB     31$                                     ; 0729
    1D            60                   06 E0 001DE 27$:     BBS     #6, (R0), 31$                           ; 0734
    03            60                   04 E1 001E2 28$:     BBC     #4, (R0), 29$                           ; 0740
                  60                   04 8A 001E6          BICB2   #4, (R0)                                ; 0742
    0A            60                   05 E1 001E9 29$:     BBC     #5, (R0), 30$                           ; 0747
            008C  C9          0090     C9 D0 001ED          MOVL    144(IRAB), 140(IRAB)                    ; 0750
                  60                   04 8A 001F4          BICB2   #4, (R0)                                ; 0751
    04            60                   02 E0 001F7 30$:     BBS     #2, (R0), 31$                           ; 0761
                              0090     C9 D4 001FB          CLRL    144(IRAB)                               ; 0763
                  60            BB     8F 8A 001FF 31$:     BICB2   #-69, (R0)                              ; 0768
                  50                   01 D0 00203          MOVL    #1, R0                                  ; 0769
                                       04 BA 00206 32$:     POPR    #^M<R2>                                 ; 0771
                                       05 00208             RSB                                             ;

; Routine Size:  521 bytes,    Routine Base:  RM$RMS3 + 0000
;  709          0772  1
```

```
1 %SBTTL  'RM$INS_ALL_SIDR'
1 GLOBAL ROUTINE RM$INS_ALL_SIDR : RL$RABREG_4567 =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Loop through all alternate key values and insert key values from
1 !       the user data record.  If update mode then only those required.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       RM$INS_ALL_SIDR()
1 !
1 ! INPUT PARAMETERS:
1 !       none
1 !
1 ! IMPLICIT INPUTS:
1 !       none
1 !
1 ! OUTPUT PARAMETERS:
1 !       none
1 !
1 ! IMPLICIT OUTPUTS:
1 !       none
1 !
1 ! ROUTINE VALUE:
1 !       none
1 !
1 ! SIDE EFFECTS:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         COMMON_RAB_STR,
2         COMMON_IO_STR,
2         R_REC_ADDR_STR,
2         R_IDX_DFN_STR;
2
2     LABEL
2         LOOP;
2
2     BUILTIN
2         AP;
2
2     ! Loop doing all of the keys NOTE:  This assumes that we are already
2     ! looking at the key 0 descriptor
2     !
2     WHILE RM$GET_NEXT_KEY()
2     DO
2
3 LOOP :  BEGIN
3
3         REC_ADDR = .RAB [ RAB$L_RBF ];
```

```
3
3          ! Check for record size large enough to contain this key value
3          !
3          IF .RAB [ RAB$W_RSZ ] LSSU .IDX_DFN [ IDX$W_MINRECSZ ]
3          THEN
3
3              ! If this key should have been updated but now record is not long
3              ! enough, the used has modified his record buffer while operating
3              ! asynchronously before the update operation is finished.
3              ! Otherwise, just leave loop so that this key value is not
3              ! inserted.
3              !
3              IF .IRAB [ IRB$V_UPDATE ]
3                  AND
3                  .BBLOCK [ .IRAB [ IRB$L_UPDBUF ] + .IDX_DFN [ IDX$B_DESC_NO ],
3                                                          UPD$V_INS_NEW ]
3              THEN
4                  RETURN RMSERR( RSZ )
3              ELSE
3                  LEAVE LOOP
3          ELSE
3
3              ! If the record fits and this is update but not marked for
3              ! insertion, then leave loop so that key is not inserted.
3              !
3              IF .IRAB [ IRB$V_UPDATE ]
3                  AND
3                  NOT .BBLOCK [ .IRAB [ IRB$L_UPDBUF ] + .IDX_DFN [ IDX$B_DESC_NO ],
3                                                          UPD$V_INS_NEW ]
3              THEN
3                  LEAVE LOOP;
3
3          ! Check that user buffer can still be read, REC_ADDR contains address
3          ! of RBF
3          !
3          IF RM$NOREAD_LONG( .RAB[RAB$W_RSZ],.REC_ADDR,.IRAB [ IRB$B_MODE ] )
3          THEN
3              RETURN RMSERR( RBF );
3
3          ! If not update, check for null key
3          !
3          IF NOT .IRAB [ IRB$V_UPDATE ]
3          THEN
4              BEGIN
4              AP = 0;     ! compare to data record, REC_ADDR points to RBF
4
4              IF NOT RM$NULLKEY( .REC_ADDR )
4              THEN
4                  LEAVE LOOP;
4
3              END;
3
3          ! Get key into key buffer 2
3          !
3          AP = 3;                 ! set up for RECORD_KEY - no overhead/expanded
3          RM$RECORD_KEY( KEYBUF_ADDR( 2 ) );
3          IRAB [ IRB$B_STOPLEVEL ] = 0;
```

```
;   825          0887  3          IRAB [ IRB$B_SPL_BITS ] = 0;
;   826          0888  3          IRAB [ IRB$W_SRCHFLAGS ] = IRB$M_POSINSERT;
;   827          0889  3          IRAB [ IRB$B_KEYSZ ] = .IDX_DFN [ IDX$B_KEYSZ ];
;   828          0890  3
;   829          0891  3          ! Define local block for status
;   830          0892  3          !
;   831          0893  4          BEGIN
;   832          0894  4
;   833          0895  4          LOCAL
;   834          0896  4              STATUS;
;   835          0897  4
;   836          0898  4          ! If key is packed decimal - check that it's a legal packed string
;   837          0899  4          !
;   838          0900  4          IF .IDX_DFN[IDX$B_DATATYPE] EQL IDX$C_PACKED
;   839          0901  4          THEN
;   840          0902  5              BEGIN
;   841          0903  5              LOCAL
;   842          0904  5                  RBF_ADDR;
;   843          0905  5
;   844          0906  5              RBF_ADDR = .RAB[RAB$L_RBF];
;   845          0907  5
;   846          0908  5              IF RM$NOREAD_LONG (.RAB[RAB$W_RSZ], .RBF_ADDR, ..RAB[IRB$B_MODE])
;   847          0909  5              THEN
;   848          0910  6                  STATUS = RMSERR( RBF )
;   849          0911  5              ELSE
;   850          0912  5                  STATUS = RM$PCKDEC_CHECK();
;   851          0913  5
;   852          0914  5              IF NOT .STATUS
;   853          0915  5              THEN
;   854          0916  6                  BEGIN
;   855          0917  6                  RM$PUTUPD_ERROR();
;   856          0918  6                  RETURN .STATUS
;   857          0919  6                  END
;   858          0920  4              END;
;   859          0921  4
;   860          0922  4          ! Insert SIDR record and do all inde  updates
;   861          0923  4          !
;   862          0924  4          STATUS = RM$INSS_OR_IDX();
;   863          0925  4
;   864          0926  5          IF .STATUS<0, 16> EQL RMSERR( IDX )
;   865          0927  4          THEN
;   866          0928  4
;   867          0929  4              ! Got an index error attempting to insert record, so make the index
;   868          0930  4              ! if error on that, clean up the alternate keys inserted so far
;   869          0931  4              ! otherwise, try to insert again
;   870          0932  4              !
;   871          0933  5              BEGIN
;   872          0934  5              STATUS = RM$MAKE_INDEX();
;   873          0935  5
;   874          0936  5              IF NOT .STATUS
;   875          0937  5              THEN
;   876          0938  6                  BEGIN
;   877          0939  6                  RM$PUTUPD_ERROR();
;   878          0940  6                  RETURN .STATUS;
;   879          0941  6
;   880          0942  6                  END
;   881          0943  5              ELSE
;  882          0944  5                STATUS = RM$INSS_OR_IDX();
;  883          0945  5
;  884          0946  4            END;
;  885          0947  4
;  886          0948  4        IF NOT .STATUS
;  887          0949  4        THEN
;  888          0950  4
;  889          0951  4            ! An error at level 0 is fatal, i.e., the record was not inserted
;  890          0952  4            ! at all otherwise just signal in index update failure
;  891          0953  4            !
;  892          0954  4            IF .IRAB [ IRB$B_STOPLEVEL ] EQL 0
;  893          0955  4            THEN
;  894          0956  5                BEGIN
;  895          0957  5                RM$PUTUPD_ERROR();
;  896          0958  5                RETURN .STATUS;
;  897          0959  5
;  898          0960  5                END
;  899          0961  4            ELSE
;  900          0962  5                BEGIN
;  901          0963  5                RAB [ RAB$L_STV ] = .STATUS;
;  902          0964  5                IRAB [ IRB$V_IDX_ERR ] = 1;
;  903          0965  4                END;
;  904          0966  4
;  905          0967  4        END                                         ! of block defining STATUS
;  906          0968  2        END;                                        ! of block LOOP
;  907          0969  2
;  908          0970  2    RETURN RMSSUC( SUC );
;  909          0971  2
;  910          0972  1    END;



                                                52  DD 00000 RM$INS_ALL_SIDR::
                                                                     PUSHL   R2                                      ; 0774
                                             0000G  30 00002 1$:     BSBW    RM$GET_NEXT_KEY                         ; 0824
                                03              50  E8 00005         BLBS    R0, 2$                                  ;
                                              00DB  31 00008         BRW     16$                                     ;
                                56       28     A8  D0 0000B 2$:     MOVL    40(RAB), REC_ADDR                       ; 0829
                       22       A7       22     A8  B1 0000F         CMPW    34(RAB), 34(IDX_DFN)                    ; 0833
                                                1B  1E 00014         BGEQU   3$                                      ;
         52        06    A9     01              03  EF 00016         EXTZV   #3, #1, 6(IRAB), R2                     ; 0842
                                E3              52  E9 0001C         BLBC    R2, 1$                                  ;
                                50       10     A7  9A 0001F         MOVZBL  16(IDX_DFN), R0                         ; 0845
                                50       64     A9  C0 00023         ADDL2   100(IRAB), R0                           ;
                                D8              60  E9 00027         BLBC    (R0), 1$                                ;
                                50     86A4     8F  3C 0002A         MOVZWL  #34468, R0                              ; 0847
                                                2C  11 0002F         BRB     5$                                      ;
         52        06    A9     01              03  EF 00031 3$:     EXTZV   #3, #1, 6(IRAB), R2                     ; 0855
                                0B              52  E9 00037         BLBC    R2, 4$                                  ;
                                50       10     A7  9A 0003A         MOVZBL  16(IDX_DFN), R0                         ; 0858
                                50       64     A9  C0 0003E         ADDL2   100(IRAB), R0                           ;
                                BD              60  E9 00042         BLBC    (R0), 1$                                ;
                                7E       0A     A9  9A 00045 4$:     MOVZBL  10(IRAB), -(SP)                         ; 0865
                                                56  DD 00049         PUSHL   REC_ADDR                                ;
                                7E       22     A8  3C 0004B         MOVZWL  34(RAB), -(SP)                          ;
```

```
                                  0000G 30 0004F        BSBW      RM$NOREAD_LONG                          ;
                           5E        0C C0 00052        ADDL2     #12, SP                                 ;
                           07        50 E9 00055        BLBC      R0, 6$                                  ;
                           50  8654  8F 3C 00058        MOVZWL    #34388, R0                              ; 0867
                                     7A 11 0005D 5$:    BRB       13$                                     ;
                           0D        52 E8 0005F 6$:    BLBS      R2, 7$                                  ; 0871
                                     5C D4 00062        CLRL      AP                                      ; 0874
                                     56 DD 00064        PUSHL     REC_ADDR                                ; 0876
                                  0000G 30 00066        BSBW      RM$NULLKEY                              ;
                           5E        04 C0 00069        ADDL2     #4, SP                                  ;
                           93        50 E9 0006C        BLBC      R0, 1$                                  ;
                           5C        03 D0 0006F 7$:    MOVL      #3, AP                                  ; 0884
                           50  00B4  CA 3C 00072        MOVZWL    180(IFAB), R0                           ; 0885
                             60 B940 9F 00077           PUSHAB    @96(IRAB)[R0]                           ;
                                  0000G 30 0007B        BSBW      RM$RECORD_KEY                           ;
                           5E        04 C0 0007E        ADDL2     #4, SP                                  ;
                               41    A9 94 00081        CLRB      65(IRAB)                                ; 0886
                               44    A9 94 00084        CLRB      68(IRAB)                                ; 0887
                      42   A9        01 B0 00087        MOVW      #1, 66(IRAB)                            ; 0888
                    00A6   C9    20  A7 90 0008B        MOVB      32(IDX_DFN), 166(IRAB)                  ; 0889
                           05    1D  A7 91 00091        CMPB      29(IDX_DFN), #5                         ; 0900
                                     24 12 00095        BNEQ      10$                                     ;
                           50    28  A8 D0 00097        MOVL      40(RAB), RBF_ADDR                       ; 0906
                           7E    0A  A9 9A 0009B        MOVZBL    10(IRAB), -(SP)                         ; 0908
                                     50 DD 0009F        PUSHL     RBF_ADDR                                ;
                           7E    22  A8 3C 000A1        MOVZWL    34(RAB), -(SP)                          ;
                                  0000G 30 000A5        BSBW      RM$NOREAD_LONG                          ;
                           5E        0C C0 000A8        ADDL2     #12, SP                                 ;
                           07        50 E9 000AB        BLBC      R0, 8$                                  ;
                           50  8654  8F 3C 000AE        MOVZWL    #34388, STATUS                          ; 0910
                                     03 11 000B3        BRB       9$                                      ;
                                  0000G 30 000B5 8$:    BSBW      RM$PCKDEC_CHECK                         ; 0912
                           1B        50 E9 000B8 9$:    BLBC      STATUS, 12$                             ; 0914
                                  0000G 30 000BB 10$:   BSBW      RM$INSS_OR_IDX                          ; 0924
                    855C   8F        50 B1 000BE        CMPW      STATUS, #34140                          ; 0926
                                     09 12 000C3        BNEQ      11$                                     ;
                                  0000G 30 000C5        BSBW      RM$MAKE_INDEX                           ; 0934
                           0B        50 E9 000C8        BLBC      STATUS, 12$                             ; 0936
                                  0000G 30 000CB        BSBW      RM$INSS_OR_IDX                          ; 0944
                           12        50 E8 000CE 11$:   BLBS      STATUS, 15$                             ; 0948
                               41    A9 95 000D1        TSTB      65(IRAB)                                ; 0954
                                     05 12 000D4        BNEQ      14$                                     ;
                                  0000G 30 000D6 12$:   BSBW      RM$PUTUPD_ERROR                         ; 0957
                                     0E 11 000D9 13$:   BRB       17$                                     ; 0958
                      0C   A8        50 D0 000DB 14$:   MOVL      STATUS, 12(RAB)                         ; 0963
                      06   A9        02 88 000DF        BISB2     #2, 6(IRAB)                             ; 0964
                                   FF1C 31 000E3 15$:   BRW       1$                                      ; 0824
                           50        01 D0 000E6 16$:   MOVL      #1, R0                                  ; 0970
                                     04 BA 000E9 17$:   POPR      #^M<R2>                                 ; 0972
                                     05 000EB           RSB                                               ;

; Routine Size:  236 bytes,    Routine Base:  RM$RMS3 + 0209


;  911          0973  1
```

```
1 %SBTTL  'RM$PUT_UPD_CHKS'
1 GLOBAL ROUTINE RM$PUT_UPD_CHKS : RL$RABREG_7 =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Perform common put/update  hecks on record size and buffer
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       RM$PUT_UPD_CHKS()
1 !
1 ! INPUT PARAMETERS:
1 !       none
1 !
1 ! IMPLICIT INPUTS:
1 !       none
1 !
1 ! OUTPUT PARAMETERS:
1 !       none
1 !
1 ! IMPLICIT OUTPUTS:
1 !       none
1 !
1 ! ROUTINE VALUE:
1 !       none
1 !
1 ! SIDE EFFECTS:
1 !       none
1 !
1 !--
1
2    BEGIN
2
2    EXTERNAL REGISTER
2        COMMON_RAB_STR,
2        R_IDX_DFN_STR;
2
2    IRAB[IRB$L_NXTBDB] = 0;
2    IRAB[IRB$V_IDX_ERR] = 0;
2    IRAB[IRB$V_RRV_ERR] = 0;
2    IRAB[IRB$V_DUP] = 0;
2    IRAB [IRB$L_RBF] = .RAB [RAB$L_RBF] ;
2    IRAB [IRB$W_RSZ] = .RAB [RAB$W_RSZ] ;
2
2    ! make sure rsz isn't greater than the maximum record size allowed
2    !
2    IF .IFAB[IFB$B_RFMORG] EQL FAB$C_FIX
2    THEN
3        BEGIN
3
3        IF .IRAB[IRB$W_RSZ] NEQU .IFAB[IFB$W_LRL]
3        THEN
4            RETURN RMSERR(RSZ)
3        END
2    ELSE
```

```
2
2              IF .IFAB[IFB$W_MRS] NEQ 0
2                  AND
2                  .IRAB[IRB$W_RSZ] GTRU .IFAB[IFB$W_MRS]
2              THEN
2                  RETURN RMSERR(RSZ);
2
2          ! set up for the primary key
2          !
2          RETURN_ON_ERROR( RM$KEY_DESC(0) );
2
2          ! make sure the record will fit in a bucket
2          !
3          BEGIN
3
3          LOCAL
3              BKT_SIZE         : WORD;
3
3          IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
3          THEN
4              BEGIN
4              BKT_SIZE = (.IDX_DFN[IDX$B_DATBKTSZ]*512) - BKT$C_OVERHDSZ - 1 -
4                      IRC$C_FIXOVHDSZ;
4
4              IF .IFAB[IFB$B_RFMORG] NEQ FAB$C_FIX
4              THEN
4                  BKT_SIZE = .BKT_SIZE - IRC$C_DATSZFLD;
4              END
3          ELSE
4              BEGIN
4              BKT_SIZE = (.IDX_DFN[IDX$B_DATBKTSZ]*512) - BKT$C_OVERHDSZ -
4                          BKT$C_DATBKTOVH - IRC$C_FIXOVHSZ3;
4
4              IF .IFAB[IFB$B_RFMORG] NEQ FAB$C_FIX
5                OR (.IFAB[IFB$B_RFMORG] EQL FAB$C_FIX
5                     AND .IDX_DFN[IDX$B_DATBKTYP] NEQU IDX$C_NCMPNCMP)
4              THEN
4                  BKT_SIZE = .BKT_SIZE - IRC$C_DATSZFLD;
4
4              IF .IDX_DFN[IDX$V_KEY_COMPR]
4              THEN
4                  BKT_SIZE = .BKT_SIZE - IRC$C_KEYCMPOVH;
4
4              IF .IDX_DFN[IDX$V_REC_COMPR]
4              THEN
4                  BKT_SIZE = .BKT_SIZE - IRC$C_DATCMPOVH;
4
3              END;
3
3          IF .IRAB[IRB$W_RSZ] GTRU .BKT_SIZE
3          THEN
3              RETURN RMSERR(RSZ);
3
2          END;
2
2          ! make sure the record is large enough to contain the whole primary key
2          !
```

```
; 1027         1088  2       IF .IRAB[IRB$W_RSZ] LSSU .IDX_DFN[IDX$W_MINRECSZ]
; 1028         1089  2       THEN
; 1029         1090  2           RETURN RMSERR(RSZ);
; 1030         1091  2
; 1031         1092  2       ! probe record buffer
; 1032         1093  2       !
; 1033         1094  2       IF RM$NOREAD_LONG(.IRAB[IRB$W_RSZ], .IRAB[IRB$L_RBF], .IRAB[IRB$B_MODE])
; 1034         1095  2       THEN
; 1035         1096  2           RETURN RMSERR(RBF);
; 1036         1097  2
; 1037         1098  3       RETURN RMSSUC(SUC)
; 1038         1099  3
; 1039         1100  1       END;



                                        3C    A9   D4 00000 RM$PUT_UPD_CHKS::
                                                                    CLRL     60(IRAB)                                ; 1013
                        05    A9      0610    8F   AA 00003         BICW2    #1552, 5(IRAB)                          ; 1015
                        58    A9        28    A8   D0 00009         MOVL     40(RAB), 88(IRAB)                       ; 1017
                        56    A9        22    A8   B0 0000E         MOVW     34(RAB), 86(IRAB)                       ; 1018
                              01        50    AA   91 00013         CMPB     80(IFAB), #1                            ; 1022
                                              09   12 00017         BNEQ     1$                                      ;
                        52    AA        56    A9   B1 00019         CMPW     86(IRAB), 82(IFAB)                      ; 1026
                                              0E   13 0001E         BEQL     2$                                      ;
                                              6D   11 00020         BRB      8$                                      ; 1028
                                        60    AA   B5 00022 1$:     TSTW     96(IFAB)                                ; 1032
                                              07   13 00025         BEQL     2$                                      ;
                        60    AA        56    A9   B1 00027         CMPW     86(IRAB), 96(IFAB)                      ; 1034
                                              61   1A 0002C         BGTRU    8$                                      ;
                                              7E   D4 0002E 2$:     CLRL     -(SP)                                   ; 1040
                                           0000G   30 00030         BSBW     RM$KEY_DESC                             ;
                              5E              04   C0 00033         ADDL2    #4, SP                                  ;
                              79              50   E9 00036         BLBC     STATUS, 11$                             ;
                              03      00B7    CA   91 00039         CMPB     183(IFAB), #3                           ; 1049
                                              17   1E 0003E         BGEQU    3$                                      ;
                              50        17    A7   9A 00040         MOVZBL   23(IDX_DFN), R0                         ; 1052
               50             50              09   78 00044         ASHL     #9, R0, R0                              ;
               51             50              16   A3 00048         SUBW3    #22, R0, BKT_SIZE                       ;
                              01        50    AA   91 0004C         CMPB     80(IFAB), #1                            ; 1055
                                              30   13 00050         BEQL     7$                                      ;
                              51              02   A2 00052         SUBW2    #2, BKT_SIZE                            ; 1057
                                              2B   11 00055         BRB      7$                                      ; 1049
                              50        17    A7   9A 00057 3$:     MOVZBL   23(IDX_DFN), R0                         ; 1061
               50             50              09   78 0005B         ASHL     #9, R0, R0                              ;
               51             50              19   A3 0005F         SUBW3    #25, R0, BKT_SIZE                       ; 1062
                              01        50    AA   91 00063         CMPB     80(IFAB), #1                            ; 1064
                                              06   12 00067         BNEQ     4$                                      ;
                              06        29    A7   91 00069         CMPB     41(IDX_DFN), #6                         ; 1066
                                              03   13 0006D         BEQL     5$                                      ;
                              51              02   A2 0006F 4$:     SUBW2    #2, BKT_SIZE                            ; 1068
               03       1C    A7              06   E1 00072 5$:     BBC      #6, 28(IDX_DFN), 6$                     ; 1070
                              51              02   A2 00077         SUBW2    #2, BKT_SIZE                            ; 1072
                                        1C    A7   95 0007A 6$:     TSTB     28(IDX_DFN)                             ; 1074
                                              03   18 0007D         BGEQ     7$                                      ;
```

```
                51          03  A2 0007F        SUBW2   #3, BKT_SIZE                    ; 1076
                51     56   A9  B1 00082 7$:    CMPW    86(IRAB), BKT_SIZE              ; 1080
                            07  1A 00086        BGTRU   8$                              ;
          22    A7     56   A9  B1 00088        CMPW    86(IRAB), 34(IDX_DFN)           ; 1088
                            06  1E 0008D        BGEQU   9$                              ;
                50   86A4   8F  3C 0008F 8$:    MOVZWL  #34468, R0                      ; 1090
                                05 00094        RSB                                     ;
                7E     0A   A9  9A 00095 9$:    MOVZBL  10(IRAB), -(SP)                 ; 1094
                       58   A9  DD 00099        PUSHL   88(IRAB)                        ;
                7E     56   A9  3C 0009C        MOVZWL  86(IRAB), -(SP)                 ;
                         0000G  30 000A0        BSBW    RM$NOREAD_LONG                  ;
                5E          0C  C0 000A3        ADDL2   #12, SP                         ;
                06          50  E9 000A6        BLBC    R0, 10$                         ;
                50   8654   8F  3C 000A9        MOVZWL  #34388, R0                      ; 1096
                                05 000AE        RSB                                     ;
                50          01  D0 000AF 10$:   MOVL    #1, R0                          ; 1098
                                05 000B2 11$:   RSB                                     ; 1100

; Routine Size:  179 bytes,    Routine Base:  RM$RMS3 + 02F5


; 1040          1101  1
```

```
%SBTTL  'RMSPUT_UPD_FIN'
GLOBAL ROUTINE RMSPUT_UPD_FIN : RL$RABREG =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       Perfrom common put/update successful completion operations
!
! CALLING SEQUENCE:
!       none
!
! INPUT PARAMETERS:
!       none
!
! IMPLICIT INPUTS:
!       none
!
! OUTPUT PARAMETERS:
!       none
!
! IMPLICIT OUTPUTS:
!       none
!
! ROUTINE VALUE:
!       none
!
! SIDE EFFECTS:
!       none
!
!--

    BEGIN

    EXTERNAL REGISTER
        COMMON_RAB_STR;

    ! All done, return information to user
    !
    RAB [ RAB$L_RFA0 ] = .IRAB[IRB$L_PUTUP_VBN];
    RAB [ RAB$W_RFA4 ] = .IRAB[IRB$W_PUTUP_ID];

    ! Return significant success codes
    !
    IF .IRAB [ IRB$V_IDX_ERR ]
    THEN
        RETURN RMSERR( OK_IDX );

    IF .IRAB [ IRB$V_DUP ]
    THEN
        RETURN RMSERR( OK_DUP );

    RETURN RMSSUC()

    END;
```

```
                         10   A8      78   A9  D0 00000 RM$PUT_UPD_FIN::
                                                          MOVL    120(IRAB), 16(RAB)                        ; 1141
                         14   A8    0080   C9  B0 00005          MOVW    128(IRAB), 20(RAB)                        ; 1142
                   06    06   A9           01  E1 0000B          BBC     #1, 6(IRAB), 1$                           ; 1146
                              50    8019   8F  3C 00010          MOVZWL  #32793, R0                                ; 1148
                                               05 00015          RSB                                               ;
                   06    05   A9           04  E1 00016 1$:      BBC     #4, 5(IRAB), 2$                           ; 1150
                              50    8011   8F  3C 0001B          MOVZWL  #32785, R0                                ; 1152
                                               05 00020          RSB                                               ;
                              50           01  D0 00021 2$:      MOVL    #1, R0                                    ; 1154
                                               05 00024          RSB                                               ; 1156

; Routine Size:  37 bytes,    Routine Base:  RM$RMS3 + 03A8


; 1097          1157  1
; 1098          1158  1 END
; 1099          1159  1
; 1100          1160  0 ELUDOM




;                                     PSECT SUMMARY
;
;       Name                        Bytes                       Attributes
;
;  RM$RMS3                               973  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      123         3        154        00:00.4



;                                     COMMAND QUALIFIERS
;
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3PUTUPD/OBJ=OBJ$:RM3PUTUPD MSRC$:RM3PUTUPD/UPDATE=(ENH$:RM3PUTUPD)

; Size:          973 code + 0 data bytes
; Run Time:         00:25.8
; Elapsed Time:     00:48.6
; Lines/CPU Min:     2699
; Lexemes/CPU-Min: 19326
; Memory Used:  235 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