C 13

```
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  SSSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSSS
```

```
_$2
Sy
---
NT$
NT$
NT$
NT$
NT$
NT$

NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$

NT$

NT$
NT$
NT$
NT$
NT$
NT$
NT$

NT$
NT$
NT$
NT$
NT$
NT$
PI
```

FILEID**RM3PUT

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

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

RM3
V04

```
0 MODULE RM3PUT (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 !                 The high level $PUT code.
1 !
1 !
1 ! ENVIRONMENT:
1 !
1 !                VAX/VMS Operating System
1 !
1 !--
1
1 !
1 ! AUTHOR:         Todd M. Katz           RE-CREATION DATE:        09-Jan-83
1 !
1 ! MODIFIED BY:
1 !
1 !       V03-022 DGB0071         Donald G. Blair         17-Jul-1984
1 !               Return STV correctly after disk-quota-exceeded error
1 !               during root bucket split.
1 !
1 !       V03-021 DAS0002         David Solomon           25-Mar-1984
1 !               Fix broken branches to RM$RU_JOURNAL3 and RM$RU_REFORMAT.
1 !
1 !       V03-020 DGB0032         Donald G. Blair         20-Mar-1984
1 !               Fix broken branch to RM$UNLOCK
```

```
1 !
1 !         V03-019 DAS0001         David Solomon           05-Mar-1984
1 !                 Don't set IRB$V_UNLOCK_RP if ULK set (FT1 QAR 628).
1 !
1 !         V03-018 TSK0001         Tamar Krichevsky        22-Jun-1983
1 !                 Fix broken branch to RM$LOCK.
1 !
1 !         V03-017 TMK0012         Todd M. Katz            26-Mar-1983
1 !                 Change the linkage for RM$RU_JOURNAL3 from RL$RABREG_467 to
1 !                 RL$RABREG_67.
1 !
1 !         V03-016 MCN0002         Maria del C. Nasr       24-Mar-1983
1 !                 More linkages reorganization.
1 !
1 !         V03-015 TMK0011         Todd M. Katz            16-Mar-1983
1 !                 Change the linkage for RM$RU_JOURNAL3 from RL$RABREG_67 to
1 !                 RL$RABREG_467.
1 !
1 !         V03-014 TMK0010         Todd M. Katz            16-Mar-1983
1 !                 Change the symbol RMSR$_PUT tp RJR$_PUT.
1 !
1 !         V03-013 TMK0009         Todd M. Katz            11-Mar-1983
1 !                 Set the bit CSH$V_LOCK in the field IRB$B_CACHEFLGS before
1 !                 calling RM$GET_BKT within RM$CONFIRM_EOF so that the bucket
1 !                 retrieved will be exclusively locked.
1 !
1 !         V03-012 MCN0001         Maria del C. Nasr       28-Feb-1983
1 !                 Reorganize linkages
1 !
1 !         V03-011 TMK0008         Todd M. Katz            19-Dec-1982
1 !                 Re-write the routines within this module adding support for
1 !                 Recovery Unit Journalling of $PUT.
1 !
1 !*****
1 !
1
1 LIBRARY 'RMSLIB:RMS';
1
1 REQUIRE 'RMSSRC:RMSIDXDEF';
1
1 ! Define default PSECTs for code.
1 !
1 PSECT
1     CODE = RM$RMS3(PSECT_ATTR),
1     PLIT = RM$RMS3(PSECT_ATTR);
1
1 ! Linkages.
1 !
1 LINKAGE
1     L_COMPARE_KEY,
1     L_ERROR_LINK1,
1     L_PRESERVE1,
1     L_QUERY_AND_LOCK,
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       ! Local Routine Linkage
1       !
1       RL$LINKAGE = JSB () :
1                       GLOBAL (COMMON_RABREG, R_IDX_DFN, R_REC_ADDR);
1
1   ! Forward Routines.
1   !
1   FORWARD ROUTINE
1       RM$UPDATE_IF            : RL$LINKAGE;
1
1   ! External Routines.
1   !
1   EXTERNAL ROUTINE
1       RM$BKT_SORT             : RL$RABREG_7,
1       RM$CLEAN_BDB            : RL$ERROR_LINK1,
1       RM$COMPARE_KEY          : RL$COMPARE_KEY,
1       RM$CSEARCH_TREE         : RL$RABREG_67,
1       RM$DELETE_UDR           : RL$RABREG_4567,
1       RM$GETBKT               : RL$RABREG_457,
1       RM$GETNEXT_REC          : RL$RABREG_67,
1       RM$INS_ALL_SIDR         : RL$RABREG_4567,
1       RM$INSERT_UDR           : RL$RABREG_4567,
1       RM$INSS_OR_IDX          : RL$RABREG_567,
1       RM$LOCK                 : RL$QUERY_AND_LOCK ADDRESSING_MODE( LONG_RELATIVE ),
1       RM$MAKE_INDEX           : RL$RABREG_7,
1       RM$PACK_REC             : RL$RABREG_567,
1       RM$PUT_UPD_CHKS         : RL$RABREG_7,
1       RM$PUT_UPD_FIN          : RL$RABREG,
1       RM$PUT_UPD_SPL          : RL$RABREG_4567,
1       RM$RECORD_KEY           : RL$PRESERVE1,
1       RM$RECORD_ID            : RL$RABREG_67,
1       RM$RECORD_VBN           : RL$PRESERVE1,
1       RM$RLSBKT               : RL$PRESERVE1,
1       RM$RU_JOURNAL3          : RL$RABREG_67 ADDRESSING_MODE( LONG_RELATIVE ),
1       RM$RU_REFORMAT          : RL$RABREG_567 ADDRESSING_MODE( LONG_RELATIVE ),
1       RM$UNLOCK               : RL$QUERY_AND_LOCK ADDRESSING_MODE( LONG_RELATIVE ),
1       RM$UPDATE3B             : RL$RABREG_67;
```

```
%SBTTL 'RM$CONFIRM_EOF'
ROUTINE RM$CONFIRM_EOF : RL$LINKAGE =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!	On position to EOF, while doing the first PUT, sequentially scan
!	the file to verify there are no non-deleted records following the
!	position of insert.
!
! CALLING SEQUENCE:
!	RM$CONFIRM_EOF()
!
! INPUT PARAMETERS:
!	NONE
!
! IMPLICIT INPUTS:
!
!	IDX_DFN			- address of the primary key descriptor
!	    IDX$B_DATBKTSZ	- size of the primary data bucket in blocks
!
!	IRAB			- address of IRAB
!	    IRB$V_ABOVELCKD	- if set, lockabove optimization is in effect
!	    IRB$L_CURBDB	- address of BDB for insertion bucket
!	    IRB$B_CUR_KREF	- current NRP key of reference
!	    IRB$L_LOCK_BDB	- address of BDB of above locked bucket
!	    IRB$L_MIDX_TMP3	- VBN of bucket containing insertion position
!
!	REC_ADDR		-  address of position of insertion
!
! OUTPUT PARAMETERS:
!	NONE
!
! IMPLICIT OUTPUTS:
!
!	IRAB			- address of IRAB
!	    IRB$V_ABOVELCKD	- 0
!	    IRB$L_CURBDB	- address of BDB for starting data bucket
!	    IRB$L_LOCK_BDB	- 0
!
!	REC_ADDR		- address of record to start re-positioning with
!
! ROUTINE VALUE:
!
!	ROP - if secondary key selected
!	SUC - if record is really EOF record
!	SEQ - is record isn't EOF record
!
! SIDE EFFECTS:
!
!	On all errors, all locked buckets are released.
!	Otherwise, RMS returns positioned to the first record in the original
!	    bucket.
!
!--
```

```
BEGIN

EXTERNAL REGISTER
    COMMON_RAB_STR,
    R_IDX_DFN_STR,
    R_REC_ADDR_STR;

GLOBAL REGISTER
    COMMON_IO_STR;

! Position to EOF not supported for secondary keys. Return the appropriate
! error.
!
IF .IRAB[IRB$B_CUR_KREF] NEQU 0
THEN
    RETURN RMSERR(ROP);

! Release the level one index bucket if lock above optimization was
! utilized.
!
IF TESTBITSC (IRAB[IRB$V_ABOVELCKD])
THEN
    RELEASE (IRAB[IRB$L_LOCK_BDB]);

! Retrieve the the address of the primary data bucket.
!
BDB = .IRAB [IRB$L_CURBDB];
BKT_ADDR = .BDB[BDB$L_ADDR];

! Scan this and subsequent primary data level buckets in the horizontal
! chain starting with the record following the point of insertion of the
! new record in order to prove that there are no records following the one
! which will be inserted.
!
DO
    BEGIN

    LOCAL
        END_OF_BUCKET,
        NEXT_BUCKET;

    ! If the current bucket is not a primary data bucket then signal a
    ! bugcheck.
    !
    IF .BKT_ADDR[BKT$B_LEVEL] NEQU 0
    THEN
        BUG_CHECK;

    ! Obtain the address of the end of the bucket.
    !
    END_OF_BUCKET = .BKT_ADDR + .BKT_ADDR[BKT$W_FREESPACE];

    ! Determine whether there are any non-deleted records in this bucket.
    !
    WHILE .REC_ADDR LSSA .END_OF_BUCKET
          AND
        NOT .REC_ADDR[IRC$V_RRV]
```

```
DO

    ! If the current record is not deleted, then released all locked
    ! buckets and return an error.
    !
    IF  NOT .REC_ADDR[IRC$V_DELETED]
          AND
        NOT .REC_ADDR[IRC$V_RU_DELETE]
    THEN
        BEGIN
        RM$CLEAN_BDB();
        RETURN RMSERR(SEQ);
        END

    ! Otherwise, position to the next record in the bucket.
    !
    ELSE
        RM$GETNEXT_REC();

! If this primary data bucket is the last bucket in the file, then
! the record RMS is inserting is not followed by any non-deleted records
! in the file, so return success.
!
IF .BKT_ADDR[BKT$V_LASTBKT]
THEN
    BEGIN

    ! If RMS had to scan multiple buckets, then it no longer has
    ! accessed the bucket containing the point of insertion of the new
    ! record. Release the current bucket, and reaccess this original
    ! bucket, so that RMS can start its re-positioning with the first
    ! record in this bucket.
    !
    IF .BDB[BDB$L_VBN] NEQU .IRAB[IRB$L_MIDX_TMP3]
    THEN
        BEGIN
        IRAB[IRB$L_CURBDB] = 0;
        RM$RLSBKT(0);

        IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
        RETURN_ON_ERROR (RM$GETBKT (.IRAB[IRB$L_MIDX_TMP3],
                                    .IDX_DFN[IDX$B_DATBKTSZ] * 512));
        IRAB[IRB$L_CURBDB] = .BDB;
        END;

    ! RMS will re-position to the point of insertion of the new record
    ! starting with the first record in the bucket.
    !
    REC_ADDR = .BKT_ADDR + BKT$C_OVERHDSZ;
    RETURN RMSSUC(SUC);
    END;

! This current bucket is not the last bucket in the file; therefore,
! save the VBN of the next primary data bucket in the horizontal chain,
! release the current bucket, access the next bucket, and continue the
! scan for non-deleted primary data records with the first record in
! that bucket.
```

```
        3          !
        3          NEXT_BUCKET = .BKT_ADDR[BKT$L_NXTBKT];
        3
        3          IRAB[IRB$L_CURBDB] = 0;
        3          RM$RLSBKT(0);
        3
        3          IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
      P 3          RETURN_ON_ERROR (RM$GETBKT (.NEXT_BUCKET,
        3                                     .IDX_DFN[IDX$B_DATBKTSZ] * 512));
        3
        3          IRAB[IRB$L_CURBDB] = .BDB;
        3          REC_ADDR = .BKT_ADDR + BKT$C_OVERHDSZ;
        3          END
        2      UNTIL 0;
        2
        2      RETURN RMSSUC(SUC);
        1      END;


                                                    .TITLE  RM3PUT
                                                    .IDENT  \V04-000\

                                                    .EXTRN  RM$BKT_SORT, RM$CLEAN_BDB
                                                    .EXTRN  RM$COMPARE_KEY, RM$CSEARCH_TREE
                                                    .EXTRN  RM$DELETE_UDR, RM$GETBKT
                                                    .EXTRN  RM$GETNEXT_REC, RM$INS_ALL_SIDR
                                                    .EXTRN  RM$INSERT_UDR, RM$INSS_OR_IDX
                                                    .EXTRN  RM$LOCK, RM$MAKE_INDEX
                                                    .EXTRN  RM$PACK_REC, RM$PUT_UPD_CHKS
                                                    .EXTRN  RM$PUT_UPD_FIN, RM$PUT_UPD_SPL
                                                    .EXTRN  RM$RECORD_KEY, RM$RECORD_ID
                                                    .EXTRN  RM$RECORD_VBN, RM$RLSBKT
                                                    .EXTRN  RM$RU_JOURNAL3, RM$RU_REFORMAT
                                                    .EXTRN  RM$UNLOCK, RM$UPDATE3B
                                                    .EXTRN  RM$BUG3

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

                               34  BB 00000 RM$CONFIRM_EOF:
                                                    PUSHR   #^M<R2,R4,R5>                          ; 0220
                      00C3     C9  95 00002         TSTB    195(IRAB)                              ; 0289
                               07  13 00006         BEQL    1$
              50      867C     8F  3C 00008         MOVZWL  #34428, R0                             ; 0291
                               46  11 0000D         BRB     6$
  11     04   A9               15  E5 0000F 1$:     BBCC    #21, 4(IRAB), 2$                       ; 0296
              54      0084     C9  D0 00014         MOVL    132(IRAB), BDB                         ; 0298
                      0084     C9  D4 00019         CLRL    132(IRAB)
                               7E  D4 0001D         CLRL    -(SP)
                     0000G     30     0001F         BSBW    RM$RLSBKT
              5E               04  C0 00022         ADDL2   #4, SP
              54        20     A9  D0 00025 2$:     MOVL    32(IRAB), BDB                          ; 0302
              55        18     A4  D0 00029         MOVL    24(BDB), BKT_ADDR                      ; 0303
                        0C     A5  95 0002D 3$:     TSTB    12(BKT_ADDR)                           ; 0320
                               03  13 00030         BEQL    4$
                     0000G     30     00032         BSBW    RM$BUG3                                ; 0321
              52        04     A5  3C 00035 4$:     MOVZWL  4(BKT_ADDR), END_OF_BUCKET             ; 0326
              52               55  C0 00039         ADDL2   BKT_ADDR, END_OF_BUCKET
```

```
                      52           56  D1 0003C 5$:    CMPL    REC_ADDR, END_OF_BUCKET              ; 0330
                                   1B  1E 0003F        BGEQU   8$                                   ;
          17          66           03  E0 00041        BBS     #3, (REC_ADDR), 8$                   ; 0332
          0E          66           02  E0 00045        BBS     #2, (REC_ADDR), 7$                   ; 0338
          0A          66           05  E0 00049        BBS     #5, (REC_ADDR), 7$                   ; 0340
                                0000G  30 0004D        BSBW    RM$CLEAN_BDB                         ; 0343
                      50     86AC  8F  3C 00050        MOVZWL  #34476, R0                           ; 0344
                                   6D  11 00055 6$:    BRB     12$                                  ;
                                0000G  30 00057 7$:    BSBW    RM$GETNEXT_REC                       ; 0350
                                   E0  11 0005A        BRB     5$                                   ; 0338
                      33       0D  A5  E9 0005C 8$:    BLBC    13(BKT_ADDR), 10$                    ; 0356
               0090   C9       1C  A4  D1 00060        CMPL    28(BDB), 144(IRAB)                   ; 0366
                                   25  13 00066        BEQL    9$                                   ;
                               20  A9  D4 00068        CLRL    32(IRAB)                             ; 0369
                                   7E  D4 0006B        CLRL    -(SP)                                ; 0370
                                0000G  30 0006D        BSBW    RM$RLSBKT                            ;
                   40         A9   01  90 00070        MOVB    #1, 64(IRAB)                         ; 0372
                      50       17  A7  9A 00074        MOVZBL  23(IDX_DFN), R0                      ; 0374
          6E          50           09  78 00078        ASHL    #9, R0, (SP)                         ;
                             0090  C9  DD 0007C        PUSHL   144(IRAB)                            ;
                                0000G  30 00080        BSBW    RM$GETBKT                            ;
                      5E           08  C0 00083        ADDL2   #8, SP                               ;
                      3B           50  E9 00086        BLBC    STATUS, 12$                          ;
                   20 A9           54  D0 00089        MOVL    BDB, 32(IRAB)                        ; 0375
                      56       0E  A5  9E 0008D 9$:    MOVAB   14(R5), REC_ADDR                     ; 0381
                                   2E  11 00091        BRB     11$                                  ; 0382
                      51       08  A5  D0 00093 10$:   MOVL    8(BKT_ADDR), NEXT_BUCKET             ; 0391
                               20  A9  D4 00097        CLRL    32(IRAB)                             ; 0393
                                   7E  D4 0009A        CLRL    -(SP)                                ; 0394
                                0000G  30 0009C        BSBW    RM$RLSBKT                            ;
                   40 A9           01  90 0009F        MOVB    #1, 64(IRAB)                         ; 0396
                      50       17  A7  9A 000A3        MOVZBL  23(IDX_DFN), R0                      ; 0398
          6E          50           09  78 000A7        ASHL    #9, R0, (SP)                         ;
                                   51  DD 000AB        PUSHL   NEXT_BUCKET                          ;
                                0000G  30 000AD        BSBW    RM$GETBKT                            ;
                      5E           08  C0 000B0        ADDL2   #8, SP                               ;
                      0E           50  E9 000B3        BLBC    STATUS, 12$                          ;
                   20 A9           54  D0 000B6        MOVL    BDB, 32(IRAB)                        ; 0400
                      56       0E  A5  9E 000BA        MOVAB   14(R5), REC_ADDR                     ; 0401
                                 FF6C  31 000BE        BRW     3$                                   ; 0310
                      50           01  D0 000C1 11$:   MOVL    #1, R0                               ; 0405
                                   34  BA 000C4 12$:   POPR    #^M<R2,R4,R5>                        ; 0406
                                       05 000C6        RSB                                          ;

; Routine Size:  199 bytes,    Routine Base:  RM$RMS3 + 0000
```

```
1 %SBTTL 'RM$PCKDEC_CHECK'
1 GLOBAL ROUTINE RM$PCKDEC_CHECK : RL$RABREG_7 =
1
1 !++
1 !
1 !     FUNCTIONAL DESCRIPTION:
1 !
1 !         This routine verifies that a packed decimal key is valid. The key to
1 !         be verified is always assumed to be in keybuffer 2.
1 !
1 !     CALLING SEQUENCE:
1 !
1 !         BSBW RM$PCKDEC_CHECK()
1 !
1 !     INPUT PARAMETERS:
1 !         NONE
1 !
1 !     IMPLICIT INPUT:
1 !
1 !         IFAB                      - address of IFAB
1 !             IFB$W_KBUFSZ          - size of one contigious keybuffer
1 !
1 !         IRAB                      - address of IRAB
1 !             IRB$L_KEYBUF          - address of the contigious keybuffers
1 !             IRB$B_KEYSZ           - size of key
1 !
1 !     OUTPUT PARAMETERS:
1 !         NONE
1 !
1 !     IMPLICIT OUTPUT:
1 !         NONE
1 !
1 !     ROUTINE_VALUE
1 !
1 !         SUC - string is a correct packed decimal string
1 !         KEY - string is not a packed decimal string
1 !
1 !     SIDE EFFECTS
1 !         NONE
1 !
1 !--
1
2       BEGIN
2
2       EXTERNAL REGISTER
2           R_IFAB_STR,
2           R_IRAB_STR;
2
2       LOCAL
2           KEYBUF   : REF BLOCK [, BYTE];
2
2       MACRO
2           LOW_MASK = 0,4,0 %,                           ! low nibble - digit (or sign)
2           HIGH_MASK = 4,4,0 %;                          ! high nibble - digit
2
2       KEYBUF = KEYBUF_ADDR(2);
2
```

```
;  402        0464  2       ! Check actual digits - each must be less than nine.
;  403        0465  2       !
;  404        0466  2       INCR I FROM 0 TO (.IRAB[IRB$B_KEYSZ] - 2) DO
;  405        0467  3           BEGIN
;  406        0468  3
;  407        0469  4           IF (.KEYBUF[.I, HIGH_MASK] GTRU 9)
;  408        0470  3               OR
;  409        0471  4              (.KEYBUF[.I, LOW_MASK] GTRU 9)
;  410        0472  3           THEN
;  411        0473  3               RETURN RMSERR(KEY);
;  412        0474  2           END;
;  413        0475  2
;  414        0476  2       ! Check last byte : sign,digit.
;  415        0477  2       !
;  416        0478  3       IF   (.KEYBUF[(.IRAB[IRB$B_KEYSZ] - 1), HIGH_MASK] GTRU 9)
;  417        0479  2            OR
;  418        0480  3          (.KEYBUF[(.IRAB[IRB$B_KEYSZ] - 1), LOW_MASK] LEQU 9)
;  419        0481  2       THEN
;  420        0482  2           RETURN RMSERR(KEY);
;  421        0483  2
;  422        0484  2       RETURN RMSSUC();
;  423        0485  2
;  424        0486  1       END;


                                       52  DD 00000 RM$PCKDEC_CHECK::
                                                       PUSHL   R2                                  ; 0408
                          50    00B4   CA  3C 00002    MOVZWL  180(IFAB), KEYBUF                   ; 0462
                          50      60   A9  C0 00007    ADDL2   96(IRAB), KEYBUF                    ;
                          52    00A6   C9  9A 0000B    MOVZBL  166(IRAB), R2                       ; 0466
                          52           02  C2 00010    SUBL2   #2, R2                              ;
                          51           01  CE 00013    MNEGL   #1, I                               ; 0469
                                       10  11 00016    BRB     2$                                  ;
        09       6140     04           04  ED 00018 1$: CMPZV  #4, #4, (I)[KEYBUF], #9             ;
                                       23  1A 0001E    BGTRU   3$                                  ;
        09       6140     04           00  ED 00020    CMPZV   #0, #4, (I)[KEYBUF], #9             ; 0471
                                       1B  1A 00026    BGTRU   3$                                  ;
                   EC     51           52  F3 00028 2$: AOBLEQ R2, I, 1$                           ; 0466
                          51    00A6   C9  9A 0002C    MOVZBL  166(IRAB), R1                       ; 0478
        09     FF A140    04           04  ED 00031    CMPZV   #4, #4, -1(R1)[KEYBUF], #9          ;
                                       09  1A 00038    BGTRU   3$                                  ;
        09     FF A140    04           00  ED 0003A    CMPZV   #0, #4, -1(R1)[KEYBUF], #9          ; 0480
                                       07  1A 00041    BGTRU   4$                                  ;
                          50    8594   8F  3C 00043 3$: MOVZWL #34196, R0                          ; 0482
                                       03  11 00048    BRB     5$                                  ;
                          50           01  D0 0004A 4$: MOVL   #1, R0                              ; 0484
                                       04  BA 0004D 5$: POPR   #^M<R2>                             ; 0486
                                           05 0004F    RSB                                         ;

; Routine Size:  80 bytes,    Routine Base:  RM$RMS3 + 00C7
```

```
%SBTTL 'RM$PUT3B'
GLOBAL ROUTINE RM$PUT3B : RL$RABREG =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       The routine directs the insertion of a new record. The steps required
!       to insert a new record are as follows:
!
!       1. Perform some operation specific validations and initializations.
!
!       2. Position to the point of insertion of the new record primary data
!          record.
!
!       3. Convert the $PUT into an $UPDATE and perform the $UPDATE if duplicate
!          primary keys are not allowed, the primary key of the new record is
!          already represented within the file, update access is allowed, and
!          updates have been requested in this particular circumstance.
!
!       4. Insert the new record into the current bucket provided there is both
!          available record IDs and sufficient free space in the bucket to
!          accomidate the new record. If there is not sufficient space or
!          or record IDs, the primary data bucket is split, the new record is
!          inserted where appropriate, and the primary index is updated as is
!          necessary.
!
!       5. All alternate keys in the new record are inserted.
!
! CALLING SEQUENCE:
!
!       BSBW RM$PUT3B()
!
! INPUT PARAMETERS:
!       NONE
!
! IMPLICIT INPUTS:
!
!       IDX_DFN                 - index descriptor for primary key
!           IDX$B_DATATYPE      - data type of primary key
!           IDX$B_DATBKTYP      - primary data bucket compression type
!           IDX$V_DUPKEYS       - if set, duplicate primary keys are allowed
!           IDX$B_KEYSZ         - size of the primary key
!
!       IFAB                    - address of IFAB
!           IFB$W_KBUFSZ        - size of a keybuffer
!           IFB$B_PLG_VER       - prologue version of file
!           IFB$B_RFMORG        - format of records
!
!       IRAB                    - address of IRAB
!           IRB$V_CON_EOF       - if set, make sure record is at end-of-file
!           IRB$L_CURBDB        - address of BDB for primary data bucket
!           IRB$V_DUP           - if set, new record is a duplicate
!           IRB$V_DUPS_SEEN     - if set, duplicate of new primary key seen
!           IRB$L_KEYBUF        - address of te contigious keybuffers
!           IRB$V_PUTS_LAST     - if set, last operation was sequential $PUT
!           IRB$W_PUTUP_ID      - RFA ID of new primary data record
```

```
!               IRB$L_PUTUP_VBN     - RFA VBN of new primary data record
!               IRB$L_RBF           - address of user record buffer
!               IRB$W_RSZ           - size of new record
!               IRB$W_UDR_ID        - RFA ID of current primary data record
!               IRB$L_UDR_VBN       - RFA VBN of current primary data record
!               IRB$V_UNLOCK_RP     - if set, there is a current record lock
!
!       RAB                         - address of RAB
!               RAB$B_RAC           - record access mode
!
! OUTPUT PARAMETERS:
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       IRAB
!               IRB$V_CON_EOF       - 0
!               IRB$V_IDX_ERR       - if set, error occurred updating index
!               IRB$L_MIDX_TMP3     - VBN of insertion primary data bucket
!               IRB$W_POS_INS       - offset of position of insertion
!               IRB$V_POSINSERT     - 1
!               IRB$V_PUTS_LAST     - if set, current operation is sequential $PUT
!               IRB$W_PUTUP_ID      - RFA ID of new primary data record
!               IRB$L_PUTUP_VBN     - RFA VBN of new primary data record
!               IRB$V_UNLOCK_RP     - if set, new record is locked
!
!       RAB                         - address of RAB
!               RAB$L_STV           - return value from RM$INSS_OR_IDX if error
!                                     occurred updating index
!
! ROUTINE VALUE:
!
!       RAC     - invalid record access mode
!       SEQ     - new records not inserted in ascending sequential order
!       SUC     - success
!
!       Also see return value from:
!
!       RM$CONFIRM_EOF
!       RM$CSEARCH_TREE
!       RM$INS_ALL_SIDR
!       RM$LOCK
!       RM$MAKE_INDEX
!       RM$PCKDEC_CHECK
!       RM$PUT_UPD_CHKS
!       RM$PUT_UPD_FIN
!       RM$PUT_UPD_SPL
!       RM$RLSBKT
!       RM$RU_JOURNAL3
!       RM$UPDATE_IF
!       RM$UPDATE3B
!
! SIDE EFFECTS:
!
!       AP is trashed.
!       Keybuffer 2 and 3 will have been trashed.
!       If the current process is within a Recovery Unit, and the file is being
```

```
1 !             Recovery Unit Journalled, then the operation is RU Journalled
1 !             before any permanent modification to the file takes place.
1 !         If the operation is a sequential $PUT, the keybuffer 6 will contain the
1 !             new record.
1 !         If the routine returns with success of any type then the new record has
1 !             been inserted.
1 !         If the routine returns with success then any required index updating has
1 !             taken place.
1 !         If the rotuine returns with a RVU error then the nwe record has been
1 !             inserted and any required index updating has probably taken place,
1 !             but some of the RRVs have probably not been correctly updated.
1 !--
1
2       BEGIN
2
2       EXTERNAL REGISTER
2           COMMON_RAB_STR;
2
2       GLOBAL REGISTER
2           R_BDB,
2           R_IDX_DFN_STR,
2           R_REC_ADDR_STR;
2
2       BUILTIN
2           AP,
2           TESTBITSC;
2
2       ! Do some validity checks and initialization, and unlock the current
2       ! primary data record if there is one to unlock.
2       !
2       IRAB[IRB$V_UPDATE] = 0;
2
2       IF TESTBITSC(IRAB[IRB$V_UNLOCK_RP])
2       THEN
2           RM$UNLOCK (.IRAB[IRB$L_UDR_VBN], .IRAB[IRB$W_UDR_ID]);
2
2       RETURN_ON_ERROR (RM$PUT_UPD_CHKS());
2
2       ! Make sure the record access is valid - only sequential and key access
2       ! is allowed.
2       !
2       ASSUME_C(RAB$C_KEY, 1);
2
2       IF .RAB[RAB$B_RAC] GTRU RAB$C_KEY
2       THEN
2           RETURN RMSERR(RAC);
2
2       ! If the record access mode is sequential and the last operation performed
2       ! was also a sequential $PUT then make sure that the key of the new record
2       ! is greater-than or equal to the key of the previous record inserted.
2       !
2       ! Move the key into keybuffer 6 and set the state bit IRB$V_PUTS_LAST for
2       ! the next operation in case it too is a sequential $PUT.
2       !
2       IF .RAB[RAB$B_RAC] EQLU RAB$C_SEQ
2       THEN
```

```
            BEGIN

            IF .IRAB[IRB$V_PUTS_LAST]
            THEN
                BEGIN

                AP = 2;
                IF  RM$COMPARE_KEY (.IRAB[IRB$L_RBF],
                                    KEYBUF_ADDR(6),
                                    .IDX_DFN[IDX$B_KEYSZ]) GTR 0
                THEN
                    RETURN RMSERR(SEQ);
                END;

            AP = 3;
            REC_ADDR = .IRAB[IRB$L_RBF];
            RM$RECORD_KEY (KEYBUF_ADDR(6));

            IRAB[IRB$V_PUTS_LAST] = 1;
            END
        ELSE
            IRAB[IRB$V_PUTS_LAST] = 0;

        ! Prepare to position to the point of insertion of the new record within the
        ! primary index.
        !
        IRAB[IRB$L_CURBDB] = 0;
        IRAB[IRB$B_STOPLEVEL] = 0;
        IRAB[IRB$B_KEYSZ] = .IDX_DFN[IDX$B_KEYSZ];
        IRAB[IRB$W_SRCHFLAGS] = IRB$M_POSINSERT;

        ! Move the record's key into keybuffer 2 and into keybuffer 3.
        !
        AP = 3;
        REC_ADDR = .IRAB[IRB$L_RBF];
        RM$RECORD_KEY (KEYBUF_ADDR(2));

        CH$MOVE (.IDX_DFN[IDX$B_KEYSZ], KEYBUF_ADDR(2), KEYBUF_ADDR(3));

        ! If the key is a packed decimal string, then call the packed decimal
        ! validation routine.
        !
        IF .IDX_DFN[IDX$B_DATATYPE] EQLU IDX$C_PACKED
        THEN
            RETURN_ON_ERROR (RM$PCKDEC_CHECK());

        ! It is possible under a very restricted set of circumstances, that
        ! multiple positionings to the point of insertion will be required. Towards
        ! this end place the positioning code within an infinite loop that can only
        ! be exited when it is firmly decided what to do with the new record. The
        ! set of circumstances under which multiple positions will be required is:
        !
        ! 1. The primary key of the new record already exists in the file.
        ! 2. The file does not allow duplicate primary keys.
        ! 3. The user has allowed updates to the file
        ! 4. The user has requested an $UPDATE to be performed in such a case.
        ! 5. The stream must wait for the record lock on the record to be updated
```

```
    !     to be granted before performing the $UPDATE.
    !
    ! In such a situation RMS must re-position to the point of insertion to
    ! determine whether it should still perform an $UPDATE, or whether that
    ! record has been removed, and it need do only a $PUT of the new record.
    !
    WHILE 1
    DO
        BEGIN

        LOCAL
            STATUS;

        ! Position to the point of insertion of the new record.
        !
        IRAB[IRB$B_SPL_BITS] = 0;
        STATUS = RM$CSEARCH_TREE();

        ! If no primary index exists then create one at this time and
        ! re-position.
        !
        IF .STATUS<0, 16> EQL RMSERR(IDX)
        THEN
            BEGIN

            RETURN_ON_ERROR (RM$MAKE_INDEX());

            STATUS = RM$CSEARCH_TREE();
            END;

        ! Terminate the current operation if RMS could not position to the point
        ! of insertion of the new record.
        !
        IF NOT .STATUS
        THEN
            RETURN .STATUS;

        ! If the IRB$V_CON_EOF state bit is set, verify that no record exists
        ! after the insert position which was just determined by
        ! RM$CSEARCH_TREE.  This will always be a best guess in a shared
        ! environment. This check is done by scanning the buckets along the
        ! horizontal chain looking for non-deleted records until the end of the
        ! chain is encountered. During this scan RMS must release the bucket
        ! containing the insertion point of the new record if it isn't the
        ! last bucket. This is because an RRV update could keep a rightward
        ! bucket locked and be trying to access this bucket. This would be a
        ! deadlock. If RMS is able to successful verify that there are no
        ! un-deleted records following the insertion point, then RMS
        ! re-positions to the point of insertion of the new record and
        ! continues.
        !
        ! Note: IRB$V_CON_EOF is turned off after the first $PUT or after any
        ! random $GET. Sequential $GETs will merely return RMS$_EOF until the
        ! IRB$V_CON_EOF bit is off.
        !
        IF .IRAB[IRB$V_CON_EOF]
        THEN
```

```
4               BEGIN
4
4               ! Save the VBN of the bucket containing the insert position.
4               ! RM$CONFIRM_EOF will use this VBN to re-position to this bucket
4               ! prior to returning if and only if there are no non-deleted records
4               ! following the current point of insertion.
4               !
4               IRAB[IRB$L_MIDX_TMP3] = .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$L_VBN];
4
4               ! Perform the check to see if there is a non-deleted record
4               ! following the current position of insertion of the new record.
4               ! If there is, return an error; otherwise, reposition to the
4               ! point of insertion for the new record and clear the state bit.
4               !
4               RETURN_ON_ERROR (RM$CONFIRM_EOF());
4               RETURN_ON_ERROR (RM$CSEARCH_TREE());
4               IRAB[IRB$V_CON_EOF] = 0;
3               END;
3
3       ! Make sure that RMS didn't see any duplicates if duplicates aren't
3       ! allowed. If RMS did see them and they are allowed, set the IRB$V_DUP
3       ! bit.
3       !
3       IF .IRAB[IRB$V_DUPS_SEEN]
3       THEN
3           IF NOT .IDX_DFN[IDX$V_DUPKEYS]
3           THEN
4               BEGIN
4
4               ! Determine whether an $UPDATE maybe done in place of the $PUT.
4               ! If so, set up do it, otherwise return the appropriate error.
4               !
5               IF NOT (STATUS = RM$UPDATE_IF())
4               THEN
4
4                   ! If there is already a record in the file with the same
4                   ! primary key as the new record, but that record has been
4                   ! RU_DELETEd by either this very same process or a process
4                   ! that has successfully completed its recovery unit then
4                   ! the new record is not a duplicate and a $PUT maybe done
4                   ! to insert it. In this case the positioning loop maybe
4                   ! left as the point of insertion of the new record has
4                   ! been located.
4                   !
4                   IF .STATUS EQLU 0
4                   THEN
4                       EXITLOOP
4
4                   ! Otherwise, some error has been encountered, so return
4                   ! the appropriate status.
4                   !
4                   ELSE
4                       RETURN .STATUS;
4
4               ! If RMS was able to lock the record to be updated without any
4               ! difficulties, and furthermore, found that it had not been
4               ! deleted within any recovery unit by this or any other process,
```

```
                        ! then go ahead and update it.
                        !
                        IF .STATUS<0,16> NEQU RMSSUC(OK_WAT)
                        THEN
                            BEGIN
                            IRAB[IRB$V_UPDATE_IF] = 1;
                            STATUS = RM$UPDATE3B();
                            IRAB[IRB$V_UPDATE_IF] = 0;
                            RETURN .STATUS;
                            END;

                        ! If RMS had to wait for a record lock to be granted on the
                        ! record which is to be updated, then while it waited it
                        ! released its lock on the primary data bucket. RMS must now
                        ! re-position back to the record, and the easiest way to do
                        ! this (taking into account that the record might have been
                        ! deleted and RMS would want to do a $PUT now) is to go through
                        ! the re-positioning loop one more time. The lock on the record
                        ! itself has already been released.
                        !
                        END

                    ! The positioning loop maybe left because the new record's primary
                    ! key has been found to be a duplicate, duplicates are allowed, and
                    ! thus the point of insertion of the new record has been located.
                    !
                    ELSE
                        BEGIN
                        IRAB[IRB$V_DUP] = 1;
                        EXITLOOP;
                        END

                ! The positioning loop maybe left because the new record's primary key
                ! is not represented in the file, and thus, the point of insertion of
                ! the new record has been located.
                !
                ELSE
                    EXITLOOP;

                END;

        ! Prepare to insert the primary data record.
        !
        BEGIN

        GLOBAL REGISTER
            COMMON_IO_STR;

        BDB = .IRAB[IRB$L_CURBDB];
        BKT_ADDR = .BDB[BDB$L_ADDR];

        ! Insert the primary data record. To do so the size of the primary data
        ! record is computed, and an attempt is made to insert it into the current
        ! primary data bucket at the point of insertion. If there is insufficient
        ! freespace in the bucket or insufficient record IDs, then a primary data
        ! bucket split is done, and the new record is inserted appropriately.
        !
```

```
4       BEGIN
4
4       LOCAL
4           ID_SPLIT,
4           RECORD_SIZE;
4
4       ID_SPLIT = 0;
4
4       ! Determine the amount of space the new record will occupy in the bucket,
4       ! and check for the availability of record IDs. If there are no record IDs
4       ! available in the bucket then the primary data bucket will have to be
4       ! split regardless of whether or not there is sufficient space in the
4       ! bucket for the new record. Both of these determinations are prologue
4       ! dependent.
4       !
4       IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
4       THEN
5           BEGIN
5           RECORD_SIZE = .IRAB[IRB$W_RSZ] + IRC$C_FIXOVHDSZ;
5
5           IF .IFAB[IFB$B_RFMORG] NEQU FAB$C_FIX
5           THEN
5               RECORD_SIZE = .RECORD_SIZE + IRC$C_DATSZFLD;
5
5           ! Check for the availability of record IDs. Because the file is a
5           ! prologue 1 or 2 file there maybe some free record IDs within the
5           ! the bucket. Free IDs occur when records are moved out of a bucket as
5           ! the result of a bucket split and the primary data bucket splitting
5           ! was not their original bucket. If free IDs are found, they maybe used.
5           !
5           IF .BKT_ADDR[BKT$B_NXTRECID] EQLU 0
5               OR
5              .BKT_ADDR[BKT$B_NXTRECID] GTRU .BKT_ADDR[BKT$B_LSTRECID]
5           THEN
5               IF NOT RM$BKT_SORT(.BKT_ADDR)
5               THEN
6                   BEGIN
6                   ID_SPLIT = 1;
6                   BKT_ADDR[BKT$B_NXTRECID] = 0;
6                   IRAB[IRB$W_POS_INS] = .REC_ADDR - .BKT_ADDR;
5                   END;
5           END
4       ELSE
5           BEGIN
5           RECORD_SIZE = RM$PACK_REC();
5           RECORD_SIZE = .RECORD_SIZE + IRC$C_FIXOVHSZ3;
5
5           ! All version 3 files have a record size, except fixed length records
5           ! that have not been compressed.
5           !
5           IF .IFAB[IFB$B_RFMORG] NEQU FAB$C_FIX
5               OR
6              (.IFAB[IFB$B_RFMORG] EQL FAB$C_FIX
6                           AND
6                       .IDX_DFN[IDX$B_DATBKTYP] NEQU IDX$C_NCMPNCMP)
5           THEN
5               RECORD_SIZE = .RECORD_SIZE + IRC$C_DATSZFLD;
```

```
	    ! Check for the availability of record IDs.
	    !
	    IF .BKT_ADDR[BKT$W_NXTRECID] EQLU 0
	    THEN
		BEGIN
		ID_SPLIT = 1;
		IRAB[IRB$W_POS_INS] = .REC_ADDR - .BKT_ADDR;
		END;
	    END;

	! If both record IDs and sufficient space are available in the bucket then
	! insert the new record; otherwise, split the primary data record inserting
	! the new record in the process.
	!
	IF  .ID_SPLIT
	     OR
	    NOT RM$INSERT_UDR (RECORD_SIZE)
	THEN
	    RETURN_ON_ERROR (RM$PUT_UPD_SPL (.RECORD_SIZE), BEGIN
							    IRAB[IRB$V_UPDATE] = 0;
							    RM$CLEAN_BDB()
							    END)

	! If RMS was able to insert the new record into the current bucket without
	! splitting it and if the operation is taking place on a RU Journallable
	! file within a Recovery Unit, then RU Journal the operation at this time
	! before the modified primary data bucket is released.
	!
	! NOTE: if a primary bucket data split was required, then the RU
	!	Journalling of this operation would have taken place within
	!	RM$PUT_UPD_SPL before the modified original bucket was released.
	!
	ELSE
	    IF .IFAB[IFB$V_RUP]
	    THEN
		BEGIN
		REC_ADDR = .IRAB[IRB$L_RBF];

		RETURN_ON_ERROR (RM$RU_JOURNAL3 (RJR$_PUT,
						 .IRAB[IRB$L_PUTUP_VBN],
						 .IRAB[IRB$W_PUTUP_ID],
						 .IRAB[IRB$W_RSZ]),
				 RM$CLEAN_BDB());
		END;
	END;

	! The new record, having been inserted, is now locked.
	!
	RETURN_ON_ERROR (RM$LOCK (.IRAB[IRB$L_PUTUP_VBN], .IRAB[IRB$W_PUTUP_ID]),
	    BEGIN
	    IRAB[IRB$V_UPDATE] = 0;
	    RM$CLEAN_BDB()
	    END);

	IF NOT .RAB[RAB$V_ULK]
	THEN
```

```
        IRAB[IRB$V_UNLOCK_RP] = 1;

    ! If the primary data bucket has split, and the primary index must be
    ! updated, then do so at this time.
    !
    IF TESTBITSC (IRAB[IRB$V_UPDATE])
    THEN
        BEGIN

        LOCAL
            STATUS;

        IRAB[IRB$B_STOPLEVEL] = 1;
        IRAB[IRB$W_SRCHFLAGS] = IRB$M_POSINSERT;

        ! If an error occurs and no previous error has been saved, save
        ! the new error status.
        !
        IF NOT (STATUS = RM$INSS_OR_IDX())
        THEN
            BEGIN
            IF .RAB[RAB$L_STV] EQL 0 THEN
                RAB[RAB$L_STV] = .STATUS OR 1^16; ! 1^16 is RMS's facility code
            IRAB[IRB$V_IDX_ERR] = 1;
            END

        END

    ! If no index update is required, then release the primary data bucket
    ! and any level above that might have been locked during positioning.
    !
    ELSE
        BEGIN

        ! Note that the original bucket may have been written with a keep lock
        ! or may not have been released at all yet, depending on the path taken
        ! to get here. This is assuming that we do not want to force a write of
        ! the bucket if no split has taken place to take advantage of deferred
        ! write. If this notion changes, so must this code.
        !
        BDB = .IRAB[IRB$L_CURBDB];
        IRAB[IRB$L_CURBDB] = 0;

        RETURN_ON_ERROR (RM$RLSBKT(0), RM$CLEAN_BDB());

        IF (BDB = .IRAB[IRB$L_LOCK_BDB]) NEQ 0
        THEN
            BEGIN
            RM$RLSBKT(0);
            IRAB[IRB$L_LOCK_BDB] = 0;
            END;
        END;

    ! Insert all of the alternate keys.
    !
    RETURN_ON_ERROR (RM$INS_ALL_SIDR(), RM$CLEAN_BDB());
```

```
   3    ! Finish up the operation.
   3    !
   3    RETURN RM$PUT_UPD_FIN();
   3
   3    END
   1    END;


                                      00FC   8F  BB 00000 RM$PUT3B::
                                                                    PUSHR   #^M<R2,R3,R4,R5,R6,R7>                  ; 0488
                          5E                  04  C2 00004           SUBL2   #4, SP
                   06     A9                  08  8A 00007           BICB2   #8, 6(IRAB)                             ; 0632
           10      04     A9                  0D  E5 0000B           BBCC    #13, 4(IRAB), 1$                        ; 0634
                          52        00BC      C9  3C 00010           MOVZWL  188(IRAB), R2                           ; 0636
                          51        00B0      C9  D0 00015           MOVL    176(IRAB), R1
                              00000000G       EF  16 0001A           JSB     RM$UNLOCK                               ;
                                      0000G   30 00020 1$:           BSBW    RM$PUT_UPD_CHKS                         ; 0638
                          3B                  50  E9 00023           BLBC    STATUS, 3$                              ;
                          01          1E      A8  91 00026           CMPB    30(RAB), #1                             ; 0645
                                              07  1B 0002A           BLEQU   2$                                      ;
                          50        8644      8F  3C 0002C           MOVZWL  #34372, R0                              ; 0647
                                              2E  11 00031           BRB     3$                                      ;
                                      1E      A8  95 00033 2$:       TSTB    30(RAB)                                 ; 0656
                                              4C  12 00036           BNEQ    5$                                      ;
           27      04     A9                  06  E1 00038           BBC     #6, 4(IRAB), 4$                         ; 0660
                          5C                  02  D0 0003D           MOVL    #2, AP                                  ; 0664
                          50        00B4      CA  3C 00040           MOVZWL  180(IFAB), R0                           ; 0666
                          50                  05  C4 00045           MULL2   #5, R0
           53             50          60      A9  C1 00048           ADDL3   96(IRAB), R0, R3
                          50          20      A7  9A 0004D           MOVZBL  32(IDX_DFN), R0                         ; 0665
                          51          58      A9  D0 00051           MOVL    88(IRAB), R1
                                      0000G   30 00055               BSBW    RM$COMPARE_KEY                          ;
                                              50  D5 00058           TSTL    R0                                      ; 0667
                                              08  15 0005A           BLEQ    4$                                      ;
                          50        86AC      8F  3C 0005C           MOVZWL  #34476, R0                              ; 0669
                                      01FA    31 00061 3$:           BRW     32$                                     ;
                          5C                  03  D0 00064 4$:       MOVL    #3, AP                                  ; 0672
                          56          58      A9  D0 00067           MOVL    88(IRAB), REC_ADDR                      ; 0673
                          50        00B4      CA  3C 0006B           MOVZWL  180(IFAB), R0                           ; 0674
                          50                  05  C4 00070           MULL2   #5, R0
                                   60 B940    9F  00073              PUSHAB  @96(IRAB)[R0]
                                      0000G   30 00077               BSBW    RM$RECORD_KEY
                          5E                  04  C0 0007A           ADDL2   #4, SP
                   04     A9          40      8F  88 0007D           BISB2   #64, 4(IRAB)                            ; 0676
                                              05  11 00082           BRB     6$                                      ; 0656
                   04     A9          40      8F  8A 00084 5$:       BICB2   #64, 4(IRAB)                            ; 0679
                                      20      A9  D4 00089 6$:       CLRL    32(IRAB)                                ; 0684
                                      41      A9  94 0008C           CLRB    65(IRAB)                                ; 0685
                 00A6     C9          20      A7  90 0008F           MOVB    32(IDX_DFN), 166(IRAB)                  ; 0686
                   42     A9                  01  B0 00095           MOVW    #1, 66(IRAB)                            ; 0687
                          5C                  03  D0 00099           MOVL    #3, AP                                  ; 0691
                          56          58      A9  D0 0009C           MOVL    88(IRAB), REC_ADDR                      ; 0692
                          50        00B4      CA  3C 000A0           MOVZWL  180(IFAB), R0                           ; 0693
                                   60 B940    9F  000A5              PUSHAB  @96(IRAB)[R0]                           ;
```

```
                                0000G 30 000A9          BSBW    RM$RECORD_KEY                           ;
                    51      20     A7 9A 000AC          MOVZBL  32(IDX_DFN), R1                         ; 0695
                    50    00B4     CA 3C 000B0          MOVZWL  180(IFAB), R0                           ;
                    6E      60  B940 3E 000B5           MOVAW   @96(IRAB)[R0], (SP)                     ;
     9E       60 B940              51 28 000BA          MOVC3   R1, @96(IRAB)[R0], @(SP)+               ;
                    05      1D     A7 91 000C0          CMPB    29(IDX_DFN), #5                         ; 0700
                                   06 12 000C4          BNEQ    7$                                      ;
                                 FEE7 30 000C6          BSBW    RM$PCKDEC_CHECK                         ; 0702
                    95             50 E9 000C9          BLBC    STATUS, 3$                              ;
                            44     A9 94 000CC 7$:      CLRB    68(IRAB)                                ; 0730
                                0000G 30 000CF          BSBW    RM$CSEARCH_TREE                         ; 0731
                    52             50 D0 000D2          MOVL    R0, STATUS                              ;
          855C      8F             52 B1 000D5          CMPW    STATUS, #34140                          ; 0736
                                   0C 12 000DA          BNEQ    8$                                      ;
                                0000G 30 000DC          BSBW    RM$MAKE_INDEX                           ; 0740
                    57             50 E9 000DF          BLBC    STATUS, 12$                             ;
                                0000G 30 000E2          BSBW    RM$CSEARCH_TREE                         ; 0742
                    52             50 D0 000E5          MOVL    R0, STATUS                              ;
                    4B             52 E9 000E8 8$:      BLBC    STATUS, 11$                             ; 0748
                            06     A9 95 000EB          TSTB    6(IRAB)                                 ; 0770
                                   1B 18 000EE          BGEQ    9$                                      ;
                    50      20     A9 D0 000F0          MOVL    32(IRAB), R0                            ; 0779
          0090      C9      1C     A0 D0 000F4          MOVL    28(R0), 144(IRAB)                       ;
                                 FDEC 30 000FA          BSBW    RM$CONFIRM_EOF                          ; 0786
                    39             50 E9 000FD          BLBC    STATUS, 12$                             ;
                                0000G 30 00100          BSBW    RM$CSEARCH_TREE                         ; 0787
                    33             50 E9 00103          BLBC    STATUS, 12$                             ;
            06      A9      80     8F 8A 00106          BICB2   #128, 6(IRAB)                           ; 0788
                            44     A9 95 0010B 9$:      TSTB    68(IRAB)                                ; 0795
                                   30 18 0010E          BGEQ    14$                                     ;
                    28      1C     A7 E8 00110          BLBS    28(IDX_DFN), 13$                        ; 0797
                                0000V 30 00114          BSBW    RM$UPDATE_IF                            ; 0804
                    52             50 D0 00117          MOVL    R0, STATUS                              ;
                    04             52 E8 0011A          BLBS    STATUS, 10$                             ;
                                   21 13 0011D          BEQL    14$                                     ; 0816
                                   15 11 0011F          BRB     11$                                     ; 0824
          8061      8F             52 B1 00121 10$:     CMPW    STATUS, #32865                          ; 0831
                                   A4 13 00126          BEQL    7$                                      ;
            06      A9             10 88 00128          BISB2   #16, 6(IRAB)                            ; 0834
                                0000G 30 0012C          BSBW    RM$UPDATE3B                             ; 0835
                    52             50 D0 0012F          MOVL    R0, STATUS                              ;
            06      A9             10 8A 00132          BICB2   #16, 6(IRAB)                            ; 0836
                    50             52 D0 00136 11$:     MOVL    STATUS, R0                              ; 0837
                                 0122 31 00139 12$:     BRW     32$                                     ;
            05      A9             10 88 0013C 13$:     BISB2   #16, 5(IRAB)                            ; 0857
                    54      20     A9 D0 00140 14$:     MOVL    32(IRAB), BDB                           ; 0877
                    55      18     A4 D0 00144          MOVL    24(BDB), BKT_ADDR                       ; 0878
                                   52 D4 00148          CLRL    ID_SPLIT                                ; 0892
                    03    00B7     CA 91 0014A          CMPB    183(IFAB), #3                           ; 0901
                                   2F 1E 0014F          BGEQU   17$                                     ;
                    6E      56     A9 3C 00151          MOVZWL  86(IRAB), RECORD_SIZE                   ; 0904
                    6E             07 C0 00155          ADDL2   #7, RECORD_SIZE                         ;
                    01      50     AA 91 00158          CMPB    80(IFAB), #1                            ; 0906
                                   03 13 0015C          BEQL    15$                                     ;
                    6E             02 C0 0015E          ADDL2   #2, RECORD_SIZE                         ; 0908
                            06     A5 95 00161 15$:     TSTB    6(BKT_ADDR)                             ; 0916
                                   07 13 00164          BEQL    16$                                     ;
```

```
                 07   A5        06   A5  91 00166        CMPB    6(BKT_ADDR), 7(BKT_ADDR)                    ; 0918
                                     38  1B 0016B        BLEQU   21$
                                     55  DD 0016D 16$:   PUSHL   BKT_ADDR                                    ; 0920
                                  0000G  30 0016F        BSBW    RM$BKT_SORT
                      5E             04  C0 00172        ADDL2   #4, SP
                      2D             50  E8 00175        BLBS    R0, 21$
                      52             01  D0 00178        MOVL    #1, ID_SPLIT                                ; 0923
                                06   A5  94 0017B        CLRB    6(BKT_ADDR)                                 ; 0924
                                     20  11 0017E        BRB     20$                                         ; 0925
                                  0000G  30 00180 17$:   BSBW    RM$PACK_REC                                 ; 0930
                      6E             50  D0 00183        MOVL    R0, RECORD_SIZE
                      6E             09  C0 00186        ADDL2   #9, RECORD_SIZE                             ; 0931
                      01        50   AA  91 00189        CMPB    80(IFAB), #1                                ; 0936
                                     06  12 0018D        BNEQ    18$
                      06        29   A7  91 0018F        CMPB    41(IDX_DFN), #6                             ; 0940
                                     03  13 00193        BEQL    19$
                      6E             02  C0 00195 18$:   ADDL2   #2, RECORD_SIZE                             ; 0942
                                06   A5  B5 00198 19$:   TSTW    6(BKT_ADDR)                                 ; 0946
                                     08  12 0019B        BNEQ    21$
                      52             01  D0 0019D        MOVL    #1, ID_SPLIT                                ; 0949
       48     A9      56             55  A3 001A0 20$:   SUBW3   BKT_ADDR, REC_ADDR, 72(IRAB)                ; 0950
                      0B             52  E8 001A5 21$:   BLBS    ID_SPLIT, 22$                               ; 0958
                                     5E  DD 001A8        PUSHL   SP                                          ; 0960
                                  0000G  30 001AA        BSBW    RM$INSERT_UDR
                      5E             04  C0 001AD        ADDL2   #4, SP
                      12             50  E8 001B0        BLBS    R0, 24$
                                     6E  DD 001B3 22$:   PUSHL   RECORD_SIZE                                 ; 0965
                                  0000G  30 001B5        BSBW    RM$PUT_UPD_SPL
                      5E             04  C0 001B8        ADDL2   #4, SP
                      2B             50  E8 001BB        BLBS    STATUS, 25$
                 06   A9             08  8A 001BE 23$:   BICB2   #8, 6(IRAB)
                                  0091  31 001C2        BRW     30$
       1E   00A2      CA             02  E1 001C5 24$:   BBC     #2, 162(IFAB), 25$                          ; 0977
                      56        58   A9  D0 001CB        MOVL    88(IRAB), REC_ADDR                          ; 0980
                      7E        56   A9  3C 001CF        MOVZWL  86(IRAB), -(SP)                             ; 0986
                      7E      0080   C9  3C 001D3        MOVZWL  128(IRAB), -(SP)
                                78   A9  DD 001D8        PUSHL   120(IRAB)
                                     13  DD 001DB        PUSHL   #19
                          00000000G  EF  16 001DD        JSB     RM$RU_JOURNAL3
                      5E             10  C0 001E3        ADDL2   #16, SP
                      6D             50  E9 001E6        BLBC    STATUS, 30$
                      52      0080   C9  3C 001E9 25$:   MOVZWL  128(IRAB), R2                               ; 0996
                      51        78   A9  D0 001EE        MOVL    120(IRAB), R1
                          00000000G  EF  16 001F2        JSB     RM$LOCK
                      C3             50  E9 001F8        BLBC    STATUS, 23$
       04             06   A8        02  E0 001FB        BBS     #2, 6(RAB), 26$                             ; 0998
                      05   A9        20  88 00200        BISB2   #32, 5(IRAB)                                ; 1000
       22             04   A9        13  E5 00204 26$:   BBCC    #19, 4(IRAB), 28$                           ; 1005
                      41   A9        01  90 00209        MOVB    #1, 65(IRAB)                                ; 1012
                      42   A9        01  B0 0020D        MOVW    #1, 66(IRAB)                                ; 1013
                                  0000G  30 00211        BSBW    RM$INSS_OR_IDX                              ; 1018
                      39             50  E8 00214        BLBS    STATUS, 29$
                                0C   A8  D5 00217        TSTL    12(RAB)                                     ; 1021
                                     09  12 0021A        BNEQ    27$
       0C     A8      50  00010000   8F  C9 0021C        BISL3   #65536, STATUS, 12(RAB)                     ; 1022
                 06   A9             02  88 00225 27$:   BISB2   #2, 6(IRAB)                                 ; 1023
                                     25  11 00229        BRB     29$                                         ; 1018
```

```
 54      20   A9 D0 0022B 28$:   MOVL    32(IRAB), BDB                   ; 1040
         20   A9 D4 0022F        CLRL    32(IRAB)                        ; 1041
              7E D4 00232        CLRL    -(SP)                           ; 1043
           0000G 30 00234        BSBW    RM$RLSBKT                       ;
 5E           04 C0 00237        ADDL2   #4, SP                          ;
 19           50 E9 0023A        BLBC    STATUS, 30$                     ;
 54    0084   C9 D0 0023D        MOVL    132(IRAB), BDB                  ; 1045
              0C 13 00242        BEQL    29$                             ;
              7E D4 00244        CLRL    -(SP)                           ; 1048
           0000G 30 00246        BSBW    RM$RLSBKT                       ;
 5E           04 C0 00249        ADDL2   #4, SP                          ;
       0084   C9 D4 0024C        CLRL    132(IRAB)                       ; 1049
           0000G 30 00250 29$:   BSBW    RM$INS_ALL_SIDR                 ; 1055
 05           50 E8 00253        BLBS    STATUS, 31$                     ;
           0000G 30 00256 30$:   BSBW    RM$CLEAN_BDB                    ;
              03 11 00259        BRB     32$                             ;
           0000G 30 0025B 31$:   BSBW    RM$PUT_UPD_FIN                  ; 1059
 5E           04 C0 0025E 32$:   ADDL2   #4, SP                          ; 1062
       00FC   8F BA 00261        POPR    #^M<R2,R3,R4,R5,R6,R7>          ;
              05 00265           RSB                                     ;

; Routine Size:  614 bytes,    Routine Base:  RM$RMS3 + 0117

; 1002         1063  1
```

```
%SBTTL 'RM$UPDATE_IF'
ROUTINE RM$UPDATE_IF : RL$LINKAGE =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       Perform all of the pre-processing necessary to convert a $PUT into
!       an $UPDATE. This includes:
!
!       1. Positioning to the record to be updated.
!       2. Locking the primary data record provided record locking has not been
!          disabled.
!       3. Determining whether this record can be updated, or whether the new
!          record should still be inserted based upon:
!
!          a. The ease with which the record lock was obtained.
!          b. The RU_DELETE status of the record which has just been locked.
!          c. Whether updates are allowed at all on the file.
!          d. Whether updates are allowed in this specific case.
!
! CALLING SEQUENCE:
!
!       BSBW RM$UPDATE_IF()
!
! INPUT PARAMETERS:
!       NONE
!
! IMPLICIT INPUTS:
!
!       IFAB                    - address of IFAB
!           IFB$V_NORECLK       - if set, record locking is disabled
!           IFB$V_UPD           - if set, updates are allowed on this file
!           IFB$B_RUP           - if set Recovery Unit is in progress
!           IFB$V_RU_RLK        - if set, perform pseudo record locking
!
!       IRAB                    - address of IRAB
!           IRB$L_CURBDB        - address of buffer descriptor for UDR's bucket
!           IRB$L_LST_REC       - address of last non-deleted record
!
!       RAB                     - address of user RAB control block
!           RAB$V_UIF           - if set, a $PUT maybe converted into $UPDATE
!
!       REC_ADDR                - address following record to be updated
!
! OUTPUT PARAMETERS:
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       IRAB                    - address of IRAB
!           IRB$W_PUTUPD_ID     - ID of primary data record to be updated
!           IRB$L_PUTUPD_VBN    - VBN of primary data record to be updated
!           IRB$V_UNLOCK_RP     - if set, release current record lock on errors
!
! ROUTINE VALUE:
!
```

```
!	0	- a $PUT is to be performed
!	RLK	- record to be updated is locked by another stream
!	DME	- dynamic memory exhausted
!	SUC	- success
!	OK_WAT	- success, but had to wait for record lock
!	OK_ALK	- success, but this stream already had this record locked
!	OK_RULK	- success, but this stream had locked this record within a RU
!
! SIDE EFFECTS:
!
!	If a decision is made to perform an $UPDATE, then the record to be
!	    updated is positioned to, made the current record, and locked
!	    (the latter if record locking is not disabled). Under certain
!	    circumstances, this record maybe reformated.
!	If a decision is made to perform a $PUT, then any record locks requested
!	    within this routine are released and this routine returns 0 with
!	    REC_ADDR pointing at where the new record is to be inserted. Under
!	    certain circumstances a record marked RU_DELETE might have been
!	    deleted from the file.
!	On any errors, or when RMS decides it must perform a re-positioning,
!	    any record locks granted within this routine are released.
!	On errors that terminate the current operation, this routine releases
!	    all bucket locks held by the stream.
!
!--

    BEGIN

    BUILTIN
	AP,
	TESTBITSC;

    EXTERNAL REGISTER
	COMMON_RAB_STR,
	R_IDX_DFN_STR,
	R_REC_ADDR_STR;

    LOCAL
	ID,
	SAVE_REC_ADDR,
	STATUS,
	VBN;

    ! Position to the preceeding record, the record which is to be updated.
    !
    SAVE_REC_ADDR = .REC_ADDR;
    REC_ADDR = .IRAB[IRB$L_LST_REC];

    ! If record locking is enabled, lock the record RMS has positioned to,
    ! otherwise, assume success for the time being.
    !
    AP = 3;

    BEGIN

    GLOBAL REGISTER
	R_BDB;
```

```
3
3           VBN = RM$RECORD_VBN();
2           END;
2
2       ID  = RM$RECORD_ID();
2
2       IF  NOT .IFAB[IFB$V_NORECLK]
2            OR
3           (.IFAB[IFB$V_RU_RLK]
3                   AND
3               .IFAB[IFB$V_RUP])
2       THEN
3           IF (STATUS = RM$LOCK (.VBN, .ID))
2               AND
3               .STATUS<0,16> NEQU RMSSUC(OK_ALK)
2           THEN
2               IRAB[IRB$V_UNLOCK_RP] = 1
2           ELSE
2               IRAB[IRB$V_UNLOCK_RP] = 0
2       ELSE
2           STATUS = RMSSUC();
2
2       ! RMS is able to lock the record it has positioned to without waiting for
2       ! the lock to be granted, but finds that this record has been deleted within
2       ! a recovery unit. In this case, RMS may proceed to insert the new record
2       ! instead of converting the current operation into an $UPDATE. Furthermore,
2       ! If the RU deleted record had been deleted within another process's
2       ! recovery unit (which has since completed successfully), RMS may now delete
2       ! this record.
2       !
2       IF  .STATUS
2            AND
3           .STATUS<0,16> NEQU RMSSUC(OK_WAT)
2            AND
2           .REC_ADDR[IRC$V_RU_DELETE]
2       THEN
3           BEGIN
3
3           ! Unless this stream already had this record locked, release the lock
3           ! that was granted above. In either case, clear the bit IRB$V_UNLOCK_RP
3           ! so no attempt will be made to release a record lock on any subsequent
3           ! errors.
3           !
3           IF TESTBITSC (IRAB[IRB$V_UNLOCK_RP])
3           THEN
3               RM$UNLOCK (.VBN, .ID);
3
3           ! If the RU deleted record has been deleted within another process's
3           ! recovery unit or within the current process's recovery unit and the
3           ! recovery unit has successfully completed, then delete the record for
3           ! good, and re-position to the new record's point of insertion.
3           !
5           IF NOT ((.STATUS<0,16> EQLU RMSSUC(OK_RULK))
4                   OR
6                   (.STATUS<0,16> EQLU RMSSUC(OK_ALK)
5                           AND
4                       .IFAB[IFB$V_RUP]))
```

```
        THEN
            BEGIN

            GLOBAL REGISTER
                R_BDB_STR,
                R_BKT_ADDR;

            LOCAL
                SAVE_REC_ID;

            SAVE_REC_ID = .REC_ADDR[IRC$W_ID];
            REC_ADDR[IRC$V_RU_DELETE] = 0;
            REC_ADDR[IRC$V_RU_UPDATE] = 0;

            BDB = .IRAB[IRB$L_CURBDB];
            RM$DELETE_UDR();
            BDB[BDB$V_DRT] = 1;

            ! If the ID of the record REC_ADDR points to is the same as the
            ! ID of the record just deleted, then the record could not be
            ! completed deleted; therefore, position past it to the point of
            ! insertion of the new record.
            !
            IF .SAVE_REC_ID EQLU .REC_ADDR[IRC$W_ID]
            THEN
                RM$GETNEXT_REC();

            END

        ! If the record which has been locked was already locked by this process
        ! (presumably within a recovery unit), then this record can not be
        ! deleted at this time. Just restore the address of the point of
        ! insertion of the new record.
        !
        ELSE
            REC_ADDR = .SAVE_REC_ADDR;

        ! Return a status of 0 to signal that a $PUT should still be done.
        !
        STATUS = 0;
        END

    ! RMS is able to lock the record it has positioned to without waiting for
    ! the lock to be granted, and furthermore finds that this record has not
    ! been deleted within a recovery unit; or, it encountered an error in its
    ! attempt to lock the record. In the former case, RMS may proceed to $UPDATE
    ! this record provided updates are allowed in this particular circumstance.
    ! In the latter case, RMS will change the error to something more
    ! appropriate if the record had been locked and updates would not have been
    ! allowed. If this routine is to return an error, all locked buckets are
    ! released at this point.
    !
    ELSE
        IF .STATUS<0,16> NEQU RMSSUC(OK_WAT)
        THEN
            BEGIN
```

```
        ! Map the current status into the appropriate error status if the
        ! file has not been opened for update access or updates are not
        ! allowed in this particular circumstance (ie - a $PUT may not be
        ! converted into an $UPDATE).
        !
        IF NOT .RAB[RAB$V_UIF]
        THEN
            STATUS = RMSERR(DUP)
        ELSE
            IF NOT .IFAB[IFB$V_UPD]
            THEN
                STATUS = RMSERR(FAC);

        ! If a $PUT maybe converted into an $UPDATE, then setup to do the
        ! $UPDATE.
        !
        IF .STATUS
        THEN
            BEGIN

            IRAB[IRB$L_PUTUP_VBN] = .VBN;
            IRAB[IRB$W_PUTUP_ID]  = .ID;

            ! Under the following set of circumstances, reformat the record
            ! which is to be updated:
            !
            ! 1. The record was put into a special format after a previous
            !    $UPDATEing within a recovery unit in order to preserve
            !    space for the duration of the recovery unit which otherwise
            !    would have been lost.
            ! 2. Either the current process is not in a recovery unit, or
            !    the current stream was able to directly obtain the record
            !    indicating the completion of the recovery unit in which
            !    the previous $UPDATE had been done.
            !
            IF .REC_ADDR[IRC$V_RU_UPDATE]
            THEN
                IF  NOT .IFAB[IFB$V_RUP]
                      OR
                    .STATUS<0,16> EQLU RMSSUC()
                THEN
                    BEGIN

                    GLOBAL REGISTER
                        R_BKT_ADDR;

                    BKT_ADDR = .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$L_ADDR];
                    RM$RU_REFORMAT();
                    END;
            END

        ! If some error has been encountered in this routine whether in
        ! locking the record to be updated or in the conversion of the $PUT
        ! into the $UPDATE, then release all bucket locks and any record
        ! locks that this routine has obtained before returning the error.
        !
        ELSE
```

```
4                   BEGIN

4                   IF TESTBITSC (IRAB[IRB$V_UNLOCK_RP])
4                   THEN
4                       RM$UNLOCK (.VBN, .ID);
4
4                   RM$CLEAN_BDB();
3                   END;
3
3               END
3
3           ! If RMS had to wait for a record lock to be granted, then while it
3           ! was waiting it had to release the bucket containing the record it
3           ! eventually was able to lock. Because anything might have happened
3           ! once this stream released the primary data bucket including the
3           ! deletion of the record RMS has locked, what RMS must do is force
3           ! a re-positioning so that it may re-determine whether it should
3           ! insert the new record or use it to update an existing record.
3           ! Before returning so that the re-positioning maybe done, RMS releases
3           ! the record lock it had to wait for.
3           !
2           ELSE
3               BEGIN
3               IRAB[IRB$V_UNLOCK_RP] = 0;
3
4               IF (STATUS = RM$UNLOCK (.VBN, .ID))
3               THEN
4                   STATUS = RMSSUC(OK_WAT)
3               ELSE
3                   RM$CLEAN_BDB();
2               END;
2
2       ! Return status.
2       !
2       RETURN .STATUS;
2
1       END;
```

```
                              3C  BB 00000 RM$UPDATE_IF:
                                               POSHR   #^M<R2,R3,R4,R5>            ; 1065
            5E                04  C2 00002     SUBL2   #4, SP                      ;
            55                56  D0 00005     MOVL    REC_ADDR, SAVE_REC_ADDR     ; 1166
            56        4C      A9  D0 00008     MOVL    76(IRAB), REC_ADDR          ; 1167
            5C                03  D0 0000C     MOVL    #3, AP                      ; 1172
                           0000G  30 0000F     BSBW    RM$RECORD_VBN               ; 1179
            54                50  D0 00012     MOVL    R0, VBN                     ;
                           0000G  30 00015     BSBW    RM$RECORD_ID                ; 1182
            52                50  D0 00018     MOVL    R0, ID                      ;
   0C      06       AA        03  E1 0001B     BBC     #3, 6(IFAB), 1$             ; 1184
   28    00A2       CA        03  E1 00020     BBC     #3, 162(IFAB), 3$           ; 1186
   22    00A2       CA        02  E1 00026     BBC     #2, 162(IFAB), 3$           ; 1188
            51                54  D0 0002C 1$: MOVL    VBN, R1                     ; 1190
               00000000G  EF  16 0002F        JSB     RM$LOCK                     ;
```

```
                        6E          50 D0 00035          MOVL    R0, STATUS                        ;
                        0D          6E E9 00038          BLBC    STATUS, 2$                        ;
              8039      8F          6E B1 0003B          CMPW    STATUS, #32825                    ; 1192
                        06          13 00040             BEQL    2$                                ;
                05      A9          20 88 00042          BISB2   #32, 5(IRAB)                      ; 1194
                        09          11 00046             BRB     4$                                ;
                05      A9          20 8A 00048  2$:     BICB2   #32, 5(IRAB)                      ; 1196
                        03          11 0004C             BRB     4$                                ; 1190
                        6E          01 D0 0004E  3$:     MOVL    #1, STATUS                        ; 1198
                        54          6E E9 00051  4$:     BLBC    STATUS, 9$                        ; 1208
              8061      8F          6E B1 00054          CMPW    STATUS, #32865                    ; 1210
                        4D          13 00059             BEQL    9$                                ;
         49             66          05 E1 0005B          BBC     #5, (REC_ADDR), 9$                ; 1212
         09     04      A9          0D E5 0005F          BBCC    #13, 4(IRAB), 5$                  ; 1221
                        51          54 D0 00064          MOVL    VBN, R1                           ; 1223
                       00000000G    EF 16 00067          JSB     RM$UNLOCK                         ;
              8071      8F          6E B1 0006D  5$:     CMPW    STATUS, #32881                    ; 1230
                        2D          13 00072             BEQL    7$                                ;
              8039      8F          6E B1 00074          CMPW    STATUS, #32825                    ; 1232
                        06          12 00079             BNEQ    6$                                ;
         20   00A2      CA          02 E0 0007B          BBS     #2, 162(IFAB), 7$                 ; 1234
                        52    01    A6 3C 00081  6$:     MOVZWL  1(REC_ADDR), SAVE_REC_ID          ; 1245
                        66    60    8F 8A 00085          BICB2   #96, (REC_ADDR)                   ; 1247
                        54    20    A9 D0 00089          MOVL    32(IRAB), BDB                     ; 1249
                             0000G  30 0008D             BSBW    RM$DELETE_UDR                     ; 1250
                0A      A4          02 88 00090          BISB2   #2, 10(BDB)                       ; 1251
 52     01      A6      10          00 ED 00094          CMPZV   #0, #16, 1(REC_ADDR), SAVE_REC_ID ; 1258
                        08          12 0009A             BNEQ    8$                                ;
                             0000G  30 0009C             BSBW    RM$GETNEXT_REC                    ; 1260
                        03          11 0009F             BRB     8$                                ; 1230
                        56          55 D0 000A1  7$:     MOVL    SAVE_REC_ADDR, REC_ADDR           ; 1270
                        6E          D4 000A4     8$:     CLRL    STATUS                            ; 1274
                        75          11 000A6             BRB     16$                               ; 1208
              8061      8F          6E B1 000A8  9$:     CMPW    STATUS, #32865                    ; 1288
                        51          13 000AD             BEQL    14$                               ;
         07     04      A8          04 E0 000AF          BBS     #4, 4(RAB), 10$                   ; 1297
                        6E    84EC  8F 3C 000B4          MOVZWL  #34028, STATUS                    ; 1299
                        0A          11 000B9             BRB     11$                               ;
         05     22      AA          03 E0 000BB  10$:    BBS     #3, 34(IFAB), 11$                 ; 1301
                        6E    8514  8F 3C 000C0          MOVZWL  #34068, STATUS                    ; 1303
                        28          6E E9 000C5  11$:    BLBC    STATUS, 13$                       ; 1308
                78      A9          54 D0 000C8          MOVL    VBN, 120(IRAB)                    ; 1312
              0080      C9          52 B0 000CC          MOVW    ID, 128(IRAB)                     ; 1313
         48             66          06 E1 000D1          BBC     #6, (REC_ADDR), 16$               ; 1327
         05   00A2      CA          02 E1 000D5          BBC     #2, 162(IFAB), 12$                ; 1329
                        01          6E B1 000DB          CMPW    STATUS, #1                        ; 1331
                        3D          12 000DE             BNEQ    16$                               ;
                50      20          A9 D0 000E0  12$:    MOVL    32(IRAB), R0                      ; 1338
                55      18          A0 D0 000E4          MOVL    24(R0), BKT_ADDR                  ;
                       00000000G    EF 16 000E8          JSB     RM$RU_REFORMAT                    ; 1339
                        2D          11 000EE             BRB     16$                               ; 1308
         25     04      A9          0D E5 000F0  13$:    BBCC    #13, 4(IRAB), 15$                 ; 1351
                        51          54 D0 000F5          MOVL    VBN, R1                           ; 1353
                       00000000G    EF 16 000F8          JSB     RM$UNLOCK                         ;
                        1A          11 000FE             BRB     15$                               ; 1355
                05      A9          20 8A 00100  14$:    BICB2   #32, 5(IRAB)                      ; 1372
                        51          54 D0 00104          MOVL    VBN, R1                           ; 1374
```

```
                 00000000G  EF  16 00107        JSB     RM$UNLOCK                    ;
            6E              50  D0 0010D        MOVL    R0, STATUS                   ;
            07              6E  E9 00110        BLBC    STATUS, 15$                  ;
            6E       8061   8F  3C 00113        MOVZWL  #32865, STATUS               ; 1376
                            03  11 00118        BRB     16$                          ;
                         0000G  30 0011A 15$:   BSBW    RM$CLEAN_BDB                 ; 1378
            50              8E  D0 0011D 16$:   MOVL    STATUS, R0                   ; 1383
                            3C  BA 00120        POPR    #^M<R2,R3,R4,R5>             ; 1385
                                05 00122        RSB                                  ;

; Routine Size:  291 bytes,    Routine Base:  RM$RMS3 + 037D

; 1326      1386  1 END
; 1327      1387  1
; 1328      1388  0 ELUDOM



;                          PSECT SUMMARY
;
;     Name                 Bytes                     Attributes
;
;  RM$RMS3                  1184  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$:RM3PUT/OBJ=OBJ$:RM3PUT MSRC$:RM3PUT/UPDATE=(ENH$:RM3PUT)

; Size:          1184 code + 0 data bytes
; Run Time:         00:31.1
; Elapsed Time:     00:53.6
; Lines/CPU Min:     2682
; Lexemes/CPU-Min: 16827
; Memory Used:  270 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