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

FILEID**RM3UPDATE

```
RRRRRRRR    MM      MM    333333    UU      UU  PPPPPPPP    DDDDDDDD      AAAAAA    TTTTTTTTTT  EEEEEEEEEE
RRRRRRRR    MM      MM    333333    UU      UU  PPPPPPPP    DDDDDDDD      AAAAAA    TTTTTTTTTT  EEEEEEEEEE
RR      RR  MMMM  MMMM  33      33  UU      UU  PP      PP  DD      DD  AA      AA      TT      EE
RR      RR  MMMM  MMMM  33      33  UU      UU  PP      PP  DD      DD  AA      AA      TT      EE
RR      RR  MM  MM  MM        33    UU      UU  PP      PP  DD      DD  AA      AA      TT      EE
RR      RR  MM  MM  MM        33    UU      UU  PP      PP  DD      DD  AA      AA      TT      EE
RRRRRRRR    MM      MM      33      UU      UU  PPPPPPPP    DD      DD  AA      AA      TT      EEEEEEEE
RRRRRRRR    MM      MM      33      UU      UU  PPPPPPPP    DD      DD  AA      AA      TT      EEEEEEEE
RR  RR      MM      MM        33    UU      UU  PP          DD      DD  AAAAAAAAAA      TT      EE
RR  RR      MM      MM        33    UU      UU  PP          DD      DD  AAAAAAAAAA      TT      EE
RR    RR    MM      MM  33    33    UU      UU  PP          DD      DD  AA      AA      TT      EE
RR    RR    MM      MM  33    33    UU      UU  PP          DD      DD  AA      AA      TT      EE
RR      RR  MM      MM    333333    UUUUUUUUUU  PP          DDDDDDDD    AA      AA      TT      EEEEEEEEEE
RR      RR  MM      MM    333333    UUUUUUUUUU  PP          DDDDDDDD    AA      AA      TT      EEEEEEEEEE

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
```

```
;     1          0001  0  ODULE RM3UPDATE (LANGUAGE (BLISS32) ,
;     2          0002  0                  IDENT = 'V04-000'
;     3          0003  0                  ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !*****************************************************************************
;     7          0007  1 !*                                                                           *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                      *
;    11          0011  1 !*                                                                           *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE     *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    17          0017  1 !*  TRANSFERRED.                                                              *
;    18          0018  1 !*                                                                           *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    21          0021  1 !*  CORPORATION.                                                              *
;    22          0022  1 !*                                                                           *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*****************************************************************************
;    28          0028  1
;    29          0029  1 !++
;    30          0030  1 !
;    31          0031  1 ! FACILITY:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !               High level update and other miscellaneous update specific
;    35          0035  1 !               routines.
;    36          0036  1 !
;    37          0037  1 !
;    38          0038  1 ! ENVIRONMENT:
;    39          0039  1 !
;    40          0040  1 !               VAX/VMS OPERATING SYSTEM
;    41          0041  1 !
;    42          0042  1 !--
;    43          0043  1
;    44          0044  1 !
;    45          0045  1 ! AUTHOR:        Todd M. Katz        RE-CREATION DATE:  22-Dec-1982
;    46          0046  1 !
;    47          0047  1 !
;    48          0048  1 ! MODIFIED BY:
;    49          0049  1 !
;    50          0050  1 !       V03-026 JWT0184         Jim Teague              21-Jun-1984
;    51          0051  1 !               Fix invalid RRV corruption problem.  On an update,
;    52          0052  1 !               if the new record is longer than the old record,
;    53          0053  1 !               a delete + insert operation is performed.  However,
;    54          0054  1 !               one thing that is different about the insert is
;    55          0055  1 !               that the new record has to have the same record
;    56          0056  1 !               id as the old one.  RMS saves the record id
;    57          0057  1 !               very early on in the update process, and never
```

```
;   58          0058  1 !               looks at it again before stuffing it into the new
;   59          0059  1 !               record.  If the desired record has been displaced
;   60          0060  1 !               by a bucket split since last_id has been saved,
;   61          0061  1 !               the record id of the new record will almost
;   62          0062  1 !               certainly be wrong.  To fix this, reevaluate
;   63          0063  1 !               last_id at a point considerably later in the
;   64          0064  1 !               update operation.
;   65          0065  1 !
;   66          0066  1 !       V03-025 JWT0183         Jim Teague              21-May-1984
;   67          0067  1 !               Fix cause of famous "RFA-accessed record deleted"
;   68          0068  1 !               error that occurred infrequently on $UPDATEs.
;   69          0069  1 !               It was simply a case of RMS not trying hard
;   70          0070  1 !               enough for the record.  RMS first tries to find the
;   71          0071  1 !               record at the last place it saw it ( LAST_ID and
;   72          0072  1 !               LAST_VBN ) by calling FIND_BY_RFA.  If someone
;   73          0073  1 !               has split the bucket out from under us, and the
;   74          0074  1 !               desired record described by the LAST fields was
;   75          0075  1 !               not in its original bucket, FIND_BY_RFA will tell
;   76          0076  1 !               us RMS$_DEL.  If that is the case, RMS will have
;   77          0077  1 !               to simply follow the record from it's RRV in the
;   78          0078  1 !               original bucket using FIND_BY_RRV.  It used to
;   79          0079  1 !               give up if FIND_BY_RFA failed with RMS$_DEL.
;   80          0080  1 !
;   81          0081  1 !       V03-024 MCN0003         Maria del C. Nasr       04-Apr-1983
;   82          0082  1 !               Change linkage of RM$NULLKEY to RL$JSB.
;   83          0083  1 !
;   84          0084  1 !       V03-023 TMK0014         Todd M. Katz            26-Mar-1983
;   85          0085  1 !               Change the linkage for RM$RU_JOURNAL3 from RL$RABREG_467 to
;   86          0086  1 !               RL$RABREG_67.
;   87          0087  1 !
;   88          0088  1 !       V03-022 MCN0002         Maria del C. Nasr       24-Mar-1983
;   89          0089  1 !               More linkages reorganization
;   90          0090  1 !
;   91          0091  1 !       V03-021 RAS0135         Ron Schaefer            17-Mar-1983
;   92          0092  1 !               Fix spelling of RJR$_UPDAT -> RJR$_UPDATE.
;   93          0093  1 !
;   94          0094  1 !       V03-020 TMK0013         Todd M. Katz            16-Mar-1983
;   95          0095  1 !               Change the linkage for RM$RU_JOURNAL3 from RL$RABREG_67 to
;   96          0096  1 !               RL$RABREG_467.
;   97          0097  1 !
;   98          0098  1 !       V03-019 TMK0012         Todd M. Katz            16-Mar-1983
;   99          0099  1 !               Change the symbol RMSR_UPDAT to RJR_UPDAT.
;  100          0100  1 !
;  101          0101  1 !       V03-018 MCN0001         Maria del C. Nasr       28-Feb-1983
;  102          0102  1 !               Reorganize linkages
;  103          0103  1 !
;  104          0104  1 !       V03-017 TMK0011         Todd M. Katz            12-Jan-1983
;  105          0105  1 !               Add support for Recovery Unit Journalling and RU ROLLBACK
;  106          0106  1 !               Recovery of ISAM files. This involves modifications to
;  107          0107  1 !               RM$REPLACE, RM$UPDATE_SCAN, and RM$UPDATE3B such that:
;  108          0108  1 !
;  109          0109  1 !                   1. The RU Journalling is done before any permanent modifications
;  110          0110  1 !                      are made to the file.
;  111          0111  1 !
;  112          0112  1 !                   2. No space is reclaimed as the result of $UPDATEs done within
;  113          0113  1 !                      Recovery Units. Inotherwards, if old SIDRs are to be deleted
;  114          0114  1 !                      they are just marked RU_DELETE, and no space is reclaimed.
```

```
;  115          0115  1 !                   Likewise, if the new primary data record is smaller that the
;  116          0116  1 !                   old record, the record is marked RU_UPDATE, it does not
;  117          0117  1 !                   shrink in size (in fact it may even grow by one byte), and
;  118          0118  1 !                   it is put into a special format so that the space maybe
;  119          0119  1 !                   reserved in case the Recovery Unit must be aborted.
;  120          0120  1 !
;  121          0121  1 !               3. If a Recovery Unit Rollback is in progress, there is no need
;  122          0122  1 !                   to insert new SIDRs which are still represented in the file
;  123          0123  1 !                   (they are marked RU_DELETE). Instead the RU_DELETE bit is
;  124          0124  1 !                   cleared in the SIDR element's control byte.
;  125          0125  1 !
;  126          0126  1 !       V03-016 TMK0010         Todd M. Katz            05-Jan-1983
;  127          0127  1 !               RM$UPDATE3B was saving, zeroing, and then restoring the current
;  128          0128  1 !               NRP key of reference while the old SIDRs were being deleted.
;  129          0129  1 !               This is no longer necessary.
;  130          0130  1 !
;  131          0131  1 !       V03-015 TMK0009         Todd M. Katz            22-Dec-1982
;  132          0132  1 !               I have changed a sufficient amount of $UPDATE, including the
;  133          0133  1 !               basic algorithms, so that the audit trail has been invalidated.
;  134          0134  1 !
;  135          0135  1 !*****
;  136          0136  1
;  137          0137  1 LIBRARY 'RMSLIB:RMS';
;  138          0138  1
;  139          0139  1 REQUIRE 'RMSSRC:RMSIDXDEF';
;  140          0204  1
;  141          0205  1 ! Define default psects for code
;  142          0206  1 !
;  143          0207  1 PSECT
;  144          0208  1     CODE = RM$RMS3(PSECT_ATTR),
;  145          0209  1     PLIT = RM$RMS3(PSECT_ATTR);
;  146          0210  1
;  147          0211  1 ! Linkages
;  148          0212  1 !
;  149          0213  1 LINKAGE
;  150          0214  1     L_COMPARE_KEY,
;  151          0215  1     L_ERROR_LINK1,
;  152          0216  1     L_ERROR_LINK2,
;  153          0217  1     L_JSB,
;  154          0218  1     L_JSB01,
;  155          0219  1     L_LINK_7_10_11,
;  156          0220  1     L_PRESERVE1,
;  157          0221  1     L_RABREG,
;  158          0222  1     L_RABREG_4567,
;  159          0223  1     L_RABREG_567,
;  160          0224  1     L_RABREG_67,
;  161          0225  1     L_RABREG_7,
;  162          0226  1     L_REC_OVHD,
;  163          0227  1
;  164          0228  1     ! Local Linkage
;  165          0229  1     !
;  166          0230  1     RL$REPLACE = JSB () :
;  167          0231  1                  GLOBAL (R_BKT_ADDR, R_IRAB, R_IFAB, R_IDX_DFN, R_REC_ADDR),
;  168          0232  1     RL$UPDATE_SCAN = JSB() :
;  169          0233  1                      GLOBAL (R_REC_ADDR, R_IMPURE, R_IDX_DFN, R_IFAB, R_IRAB, R_RAB);
;  170          0234  1
;  171          0235  1 ! External Routines
```

```
;  172     0236  1 !
;  173     0237  1 EXTERNAL ROUTINE
;  174     0238  1     RM$CLEAN_BDB        : RL$ERROR_LINK1 NOVALUE,
;  175     0239  1     RM$COMPARE_KEY      : RL$COMPARE_KEY,
;  176     0240  1     RM$DELETE_SIDR      : RL$RABREG_7,
;  177     0241  1     RM$EXPAND_KEYD      : RL$JSB01,
;  178     0242  1     RM$FIND_BY_RFA      : RL$RABREG_67,
;  179     0243  1     RM$FIND_BY_RRV      : RL$RABREG_67,
;  180     0244  1     RM$GET_NEXT_KEY     : RL$LINK_7_10_11,
;  181     0245  1     RM$INS_ALL_SIDR     : RL$RABREG_4567,
;  182     0246  1     RM$INSERT_ODR       : RL$RABREG_4567,
;  183     0247  1     RM$INSS_OR_IDX      : RL$RABREG_567,
;  184     0248  1     RM$KEY_DESC         : RL$RABREG_7,
;  185     0249  1     RM$MOVE             : RL$PRESERVE1,
;  186     0250  1     RM$NULLKEY          : RL$JSB,
;  187     0251  1     RM$PACK_REC         : RL$RABREG_567,
;  188     0252  1     RM$PUT_UPD_CHKS     : RL$RABREG_7,
;  189     0253  1     RM$PUTUPD_ERROR     : RL$ERROR_LINK2 NOVALUE,
;  190     0254  1     RM$PUT_UPD_FIN      : RL$RABREG,
;  191     0255  1     RM$PUT_UPD_SPL      : RL$RABREG_4567,
;  192     0256  1     RM$RECORD_ID        : RL$RABREG_67,
;  193     0257  1     RM$RECORD_KEY       : RL$PRESERVE1,
;  194     0258  1     RM$RECORD_VBN       : RL$PRESERVE1,
;  195     0259  1     RM$REC_OVHD         : RL$REC_OVHD,
;  196     0260  1     RM$RLSBKT           : RL$PRESERVE1,
;  197     0261  1     RM$RU_JOURNAL3      : RL$RABREG_67,
;  198     0262  1     RM$UNPACK_REC       : RL$JSB01,
;  199     0263  1     RM$UPDDELCOM        : RL$RABREG_67;
```

```
;  201     0264  1 %SBTTL  'RM$REPLACE'
;  202     0265  1 ROUTINE RM$REPLACE (NEW_REC_SIZE) : RL$REPLACE =
;  203     0266  1
;  204     0267  1 !++
;  205     0268  1 !
;  206     0269  1 ! FUNCTIONAL DESCRIPTION:
;  207     0270  1 !
;  208     0271  1 ! This routine updates the record pointed to by REC_ADDR if the new record is
;  209     0272  1 ! the same size or smaller than record being updated. Otherwise, this routine
;  210     0273  1 ! deletes entirely the old record thus setting up the conditions for performing
;  211     0274  1 ! a bucket split.
;  212     0275  1 !
;  213     0276  1 ! If the $UPDATE is taking place within a Recovery Unit and the new record is
;  214     0277  1 ! smaller then the old record by more than one byte then the record is marked
;  215     0278  1 ! RU_UPDATE and placed into a special format so that the old size maybe reserved
;  216     0279  1 ! in case it become necessary to ROLLBACK recover the Recovery Unit. This
;  217     0280  1 ! special format is to have the size of the amount of space that is being
;  218     0281  1 ! reserved in the primary data record overhead's record size field while the
;  219     0282  1 ! true size of the record occupies the last two bytes of this reserved space.
;  220     0283  1 !
;  221     0284  1 ! If the $UPDATE is taking place within a Recovery Unit and the new record is
;  222     0285  1 ! smaller that the old record by exactly one byte, then the primary data record
;  223     0286  1 ! must also be marked RU_UPDATE and placed into the special format. However, as
;  224     0287  1 ! placing the record in a special format would require it to grow in size by one
;  225     0288  1 ! byte, this routine does not replace the old with the new record. Instead, the
;  226     0289  1 ! old record is deleted, the state bit IRB$V_RU_UPDATE is set, and a 0 is
;  227     0290  1 ! returned indicating that a strict replacement could not be done. When the new
;  228     0291  1 ! record is eventually inserted, the state bit IRB$V_RU_UPDATE indicates that
;  229     0292  1 ! its size must be increased by two, and that it should be put into this special
;  230     0293  1 ! format.
;  231     0294  1 !
;  232     0295  1 ! If the $UPDATE is taking place as a RU ROLLBACK Recovery operation and the
;  233     0296  1 ! new record is smaller than the old record by more than one byte, then the
;  234     0297  1 ! primary data record is also marked RU_UPDATE and is placed into the special
;  235     0298  1 ! format. This special formatting is not necessary if the new record is exactly
;  236     0299  1 ! one byte smaller than the old record.
;  237     0300  1 !
;  238     0301  1 ! CALLING SEQUENCE:
;  239     0302  1 !
;  240     0303  1 !       RM$REPLACE()
;  241     0304  1 !
;  242     0305  1 ! INPUT PARAMETERS:
;  243     0306  1 !
;  244     0307  1 !       NEW_REC_SIZE    - size of the new record
;  245     0308  1 !
;  246     0309  1 ! IMPLICIT INPUTS:
;  247     0310  1 !
;  248     0311  1 !       BKT_ADDR                - address of the primary data bucket
;  249     0312  1 !           BKT$W_FREESPACE     - freespace offset pointer
;  250     0313  1 !
;  251     0314  1 !       IDX_DFN                 - address of the primary key index descriptor
;  252     0315  1 !           IDX$V_KEY_COMPR     - if set, primary key compression is enabled
;  253     0316  1 !
;  254     0317  1 !       IFAB                    - address of the IFAB
;  255     0318  1 !           IFB$W_KBUFSZ        - size of a keybuffer
;  256     0319  1 !           IFB$B_PLG_VER       - prologue version of the file
;  257     0320  1 !           IFB$V_RUP           - if set, Recovery Unit is in progress
```

```
;  258     0321  1 !           IFB$V_RU_RECVR      - if set, RU ROLLBACK Recovery in progress
;  259     0322  1 !
;  260     0323  1 !       IRAB                    - address of the IRAB
;  261     0324  1 !           IRB$L_KEYBUF        - address of the contigious keybuffers
;  262     0325  1 !           IRB$L_RECBUF        - address of the record buffer
;  263     0326  1 !           IPB$L_RBF           - address of the user's record buffer
;  264     0327  1 !
;  265     0328  1 !       REC_ADDR                - address of the primary data record
;  266     0329  1 !
;  267     0330  1 ! OUTPUT PARAMETERS:
;  268     0331  1 !       NONE
;  269     0332  1 !
;  270     0333  1 ! IMPLICIT OUTPUTS:
;  271     0334  1 !
;  272     0335  1 !       BKT_ADDR                - address of the primary data bucket
;  273     0336  1 !           BKT$W_FREESPACE     - freespace offset pointer
;  274     0337  1 !
;  275     0338  1 !       IRAB                    - address of IRAB
;  276     0339  1 !           IRB$V_RU_UPDATE     - if set, build new record in special format
;  277     0340  1 !
;  278     0341  1 ! ROUTINE VALUE:
;  279     0342  1 !
;  280     0343  1 !       0       - if the new record did not replace the old record
;  281     0344  1 !       1       - if the new record replaced the old record
;  282     0345  1 !
;  283     0346  1 ! SIDE EFFECTS:
;  284     0347  1 !
;  285     0348  1 !       AP and keybuffer 5 are trashed.
;  286     0349  1 !
;  287     0350  1 !       If STATUS = 0, then the old primary data record has been deleted, the
;  288     0351  1 !           bucket's freespace offset pointer has been updated to reflect this,
;  289     0352  1 !           and keybuffer 3 contains the primary key of the record.
;  290     0353  1 !       If STATUS = 1, then the old primary data record has been replaced by the
;  291     0354  1 !           new record, and any difference in size between the two is reflected
;  292     0355  1 !           both within the record's size field, and in the bucket's freespace
;  293     0356  1 !           offset pointer.
;  294     0357  1 !
;  295     0358  1 !--
;  296     0359  1
;  297     0360  2     BEGIN
;  298     0361  2
;  299     0362  2     EXTERNAL REGISTER
;  300     0363  2         R_BKT_ADDR_STR,
;  301     0364  2         R_IDX_DFN_STR,
;  302     0365  2         R_IFAB_STR,
;  303     0366  2         R_IRAB_STR,
;  304     0367  2         R_REC_ADDR_STR;
;  305     0368  2
;  306     0369  2     GLOBAL REGISTER
;  307     0370  2         R_RAB,
;  308     0371  2         R_IMPURE,
;  309     0372  2         R_BDB;
;  310     0373  2
;  311     0374  2     BUILTIN
;  312     0375  2         AP;
;  313     0376  2
;  314     0377  2     LOCAL
```

```
;  315     0378  2          DIFFERENCE      : SIGNED,
;  316     0379  2          RECORD_OVHD,
;  317     0380  2          STATUS,
;  318     0381  2          OLD_REC_SIZE,
;  319     0382  2          OLD_REC_ADDR;
;  320     0383  2
;  321     0384  2      ! Retrive the size of the old record.
;  322     0385  2      !
;  323     0386  3      BEGIN
;  324     0387  3
;  325     0388  3      LOCAL
;  326     0389  3          REC_SIZE;
;  327     0390  3
;  328     0391  3      RECORD_OVHD = RM$REC_OVHD(0; REC_SIZE);
;  329     0392  3      OLD_REC_SIZE = .REC_SIZE;
;  330     0393  3      OLD_REC_ADDR = .REC_ADDR + .RECORD_OVHD;
;  331     0394  2      END;
;  332     0395  2
;  333     0396  2      ! If the key is compressed, save in key buffer 5 the key of the record
;  334     0397  2      ! that will be deleted, including the compression overhead.  This might
;  335     0398  2      ! be used by RM$EXPAND_KEYD later.
;  336     0399  2      !
;  337     0400  2      IF .IDX_DFN[IDX$V_KEY_COMPR]
;  338     0401  2      THEN
;  339     0402  2          RM$MOVE (.(.OLD_REC_ADDR)<0,8> + 2, .OLD_REC_ADDR, KEYBUF_ADDR(5));
;  340     0403  2
;  341     0404  2      ! Figure out what the difference in records sizes is.
;  342     0405  2      !
;  343     0406  2      DIFFERENCE = .OLD_REC_SIZE - .NEW_REC_SIZE;
;  344     0407  2
;  345     0408  2      ! If the record will grow in size as a result of being updated, setup
;  346     0409  2      ! so the old record can be removed entirely from the bucket and for
;  347     0410  2      ! the split of the primary data bucket in case it becomes necessary.
;  348     0411  2      !
;  349     0412  2      ! If the record will shrink in size by one byte as a result of being updated
;  350     0413  2      ! and the process is currently in a Recovery Unit, then also setup so that
;  351     0414  2      ! the old record can be removed entirely from the bucket. In addition, set
;  352     0415  2      ! the state bit IRB$V_RU_UPDATE so that when this record gets built, it
;  353     0416  2      ! will be built in the special format that is required.
;  354     0417  2      !
;  355     0418  2      IF   .DIFFERENCE LSS 0
;  356     0419  2           OR
;  357     0420  3          (.IFAB[IFB$V_RUP]
;  358     0421  3                    AND
;  359     0422  3                   .DIFFERENCE EQL 1)
;  360     0423  2      THEN
;  361     0424  3          BEGIN
;  362     0425  3
;  363     0426  3          IF .IFAB[IFB$V_RUP]
;  364     0427  3              AND
;  365     0428  3              .DIFFERENCE EQL 1
;  366     0429  3          THEN
;  367     0430  3              IRAB[IRB$V_RU_UPDATE] = 1;
;  368     0431  3
;  369     0432  3          DIFFERENCE = .OLD_REC_ADDR + .OLD_REC_SIZE - .REC_ADDR;
;  370     0433  3          OLD_REC_ADDR = .REC_ADDR;
;  371     0434  3
```

```
;  372     0435  3          ! Get primary key value into keybuffer 3 for use by RM$PUT_UPD_SPL.
;  373     0436  3          !
;  374     0437  3          REC_ADDR = .IRAB[IRB$L_RBF];
;  375     0438  3
;  376     0439  3          AP = 3;
;  377     0440  3          RM$RECORD_KEY(KEYBUF_ADDR(3));
;  378     0441  3
;  379     0442  3          STATUS = 0;
;  380     0443  3          REC_ADDR = .OLD_REC_ADDR;
;  381     0444  3          END
;  382     0445  3
;  383     0446  3      ! The new record will be able to fit into the space currently occupied
;  384     0447  3      ! by the old record either because the records are the same size, or the
;  385     0448  3      ! new record is smaller than the old record.
;  386     0449  3      !
;  387     0450  2      ELSE
;  388     0451  3          BEGIN
;  389     0452  3
;  390     0453  3          LOCAL
;  391     0454  3              BUFFER;
;  392     0455  3
;  393     0456  3          ! Assume that the record will not be in special format and clear the
;  394     0457  3          ! appropriate record control bit.
;  395     0458  3          !
;  396     0459  3          REC_ADDR[IRC$V_RU_UPDATE] = 0;
;  397     0460  3
;  398     0461  3          ! The new record is smaller then the old record, so fix the record size.
;  399     0462  3          !
;  400     0463  3          IF .DIFFERENCE NEQ 0
;  401     0464  3          THEN
;  402     0465  3
;  403     0466  3              ! If the process is not currently within a Recovery Unit or the
;  404     0467  3              ! operation is not a RU ROLLBACK Recovery operation; or, if the
;  405     0468  3              ! current operation is a RU ROLLBACK Recovery operation but the
;  406     0469  3              ! record's size would only decrease by one byte, then just move
;  407     0470  3              ! in the new size into the record overhead's size field.
;  408     0471  3              !
;  409     0472  4              IF  NOT (.IFAB[IFB$V_RUP]
;  410     0473  4                       OR
;  411     0474  4                      .IFAB[IFB$V_RU_RECVR])
;  412     0475  3                   OR
;  413     0476  4                  (.IFAB[IFB$V_RU_RECVR]
;  414     0477  4                       AND
;  415     0478  4                      .DIFFERENCE EQLU 1)
;  416     0479  3              THEN
;  417     0480  3                  (.OLD_REC_ADDR - IRC$C_DATSZFLD)<0, 16> = .NEW_REC_SIZE
;  418     0481  3
;  419     0482  3              ! If the process is currently within a Recovery Unit then leave the
;  420     0483  3              ! record size field in the record overhead alone, and instead move
;  421     0484  3              ! the new size into the last two bytes in the record's reserved
;  422     0485  3              ! space. Set the RU_UPDATE bit in the record's control byte to
;  423     0486  3              ! indicate that the record is in a special format, and zero
;  424     0487  3              ! DIFFERENCE so that no space is reclaimed from the record.
;  425     0488  3              !
;  426     0489  3              ! This sequence is also done if the current operation is a RU
;  427     0490  3              ! ROLLBACK Recovery operation and the record shrinks by more than
;  428     0491  3              ! one byte.
```

```
;  429     0492  3              !
;  430     0493  3              ELSE
;  431     0494  4                  BEGIN
;  432     0495  4                  REC_ADDR[IRC$V_RU_UPDATE] = 1;
;  433     0496  4                  DIFFERENCE = 0;
;  434     0497  5                  (.OLD_REC_ADDR + .OLD_REC_SIZE
;  435     0498  4                                 - IRC$C_DATSZFLD)<0,16> = .NEW_REC_SIZE;
;  436     0499  3                  END;
;  437     0500  3
;  438     0501  3          ! Move in the new record. Where the new record is moved in from is
;  439     0502  3          ! prologue dependent.
;  440     0503  3          !
;  441     0504  3          IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  442     0505  3          THEN
;  443     0506  3              BUFFER = .IRAB[IRB$L_RBF]
;  444     0507  3          ELSE
;  445     0508  3              BUFFER = .IRAB[IRB$L_RECBUF];
;  446     0509  3
;  447     0510  3          OLD_REC_ADDR = RM$MOVE (.NEW_REC_SIZE, .BUFFER, .OLD_REC_ADDR);
;  448     0511  3
;  449     0512  3          STATUS = 1;
;  450     0513  2          END;
;  451     0514  2
;  452     0515  2      ! If the new record grew in size delete it from the bucket and fix up the
;  453     0516  2      ! bucket's freespace offset pointer. If the record shrunk in size, then
;  454     0517  2      ! shuffle all the trailing records in the bucket to take up this available
;  455     0518  2      ! space and fix up the bucket's freespace offset pointer.
;  456     0519  2      !
;  457     0520  3      BEGIN
;  458     0521  3
;  459     0522  3      LOCAL
;  460     0523  3          LENGTH;
;  461     0524  3
;  462     0525  3      IF .DIFFERENCE NEQ 0
;  463     0526  3      THEN
;  464     0527  4          BEGIN
;  465     0528  4
;  466     0529  4          BKT_ADDR[BKT$W_FREESPACE] = .BKT_ADDR[BKT$W_FREESPACE]
;  467     0530  4                                    - .DIFFERENCE<0, 16>;
;  468     0531  4
;  469     0532  5          LENGTH = .BKT_ADDR[BKT$W_FREESPACE] - (.OLD_REC_ADDR
;  470     0533  4                                               - .BKT_ADDR)<0, 16>;
;  471     0534  4
;  472     0535  4          IF .LENGTH GTR 0
;  473     0536  4          THEN
;  474     0537  4              RM$MOVE (.LENGTH, (.OLD_REC_ADDR + .DIFFERENCE), .OLD_REC_ADDR);
;  475     0538  4
;  476     0539  3          END;
;  477     0540  3
;  478     0541  3      ! If STATUS is 0, then the record grew in size, it has been deleted from
;  479     0542  3      ! the bucket, and a regular record insert must be done. If the key is
;  480     0543  3      ! compressed, the key of the following record (if there is one) must be
;  481     0544  3      ! expanded to reflect the deleted record.
;  482     0545  3      !
;  483     0546  3      IF   .IDX_DFN[IDX$V_KEY_COMPR]
;  484     0547  3           AND
;  485     0548  3           .STATUS EQLU 0
```

```
;  486          0549  3          AND
;  487          0550  3          .LENGTH GTR 0
;  488          0551  3          AND
;  489          0552  3          NOT .REC_ADDR[IRC$V_RRV]
;  490          0553  3      THEN
;  491          0554  3          RM$EXPAND_KEYD (KEYBUF_ADDR(5), .REC_ADDR + .RECORD_OVHD);
;  492          0555  3
;  493          0556  2      END;
;  494          0557  2
;  495          0558  2      RETURN .STATUS;
;  496          0559  2
;  497          0560  1      END;
```

```
                                        .TITLE  RM3UPDATE
                                        .IDENT  \V04-000\

                                        .EXTRN  RM$CLEAN_BDB, RM$COMPARE_KEY
                                        .EXTRN  RM$DELETE_SIDR, RM$EXPAND_KEYD
                                        .EXTRN  RM$FIND_BY_RFA, RM$FIND_BY_RRV
                                        .EXTRN  RM$GET_NEXT_KEY
                                        .EXTRN  RM$INS_ALL_SIDR
                                        .EXTRN  RM$INSERT_IDR, RM$INSS_OR_IDX
                                        .EXTRN  RM$KEY_DESC, RM$MOVE
                                        .EXTRN  RM$NULLKEY, RM$PACK_REC
                                        .EXTRN  RM$PUT_UPD_CHKS
                                        .EXTRN  RM$PUTUPD_ERROR
                                        .EXTRN  RM$PUT_UPD_FIN, RM$PUT_UPD_SPL
                                        .EXTRN  RM$RECORD_ID, RM$RECORD_KEY
                                        .EXTRN  RM$RECORD_VBN, RM$REC_OVHD
                                        .EXTRN  RM$RLSBKT, RM$RU_JOURNAL3
                                        .EXTRN  RM$UNPACK_REC, RM$UPDDELCOM

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

                        091C    8F  BB 00000 RM$REPLACE:
                                            PUSHR   #^M<R2,R3,R4,R8,R11>                        ; 0265
                     5E                  08  C2 00004                SUBL2   #8, SP
                                         51  D4 00007                CLRL    R1                  ; 0391
                                   0000G 30  00009                BSBW    RM$REC_OVHD
                     6E                  50  D0 0000C                MOVL    R0, RECORD_OVHD
                     53                  51  D0 0000F                MOVL    REC_SIZE, OLD_REC_SIZE  ; 0392
                51   56                  6E  C1 00012                ADDL3   RECORD_OVHD, REC_ADDR, OLD_REC_ADDR ; 0393
                17   1C             A7   06  E1 00016                BBC     #6, 28(IDX_DFN), 1$ ; 0400
                     50        00B4      CA  3C 0001B                MOVZWL  180(IFAB), R0       ; 0402
                               60 B940     DF  00020                PUSHAL  @96(IRAB)[R0]
                                           51  DD 00024                PUSHL   OLD_REC_ADDR
                     7E                    61  9A 00026                MOVZBL  (OLD_REC_ADDR), -(SP)
                     6E                    02  C0 00029                ADDL2   #2, (SP)
                                     0000G 30  0002C                BSBW    RM$MOVE
                     5E                    0C  C0 0002F                ADDL2   #12, SP
                52             53      20   AE  C3 00032 1$:           SUBL3   NEW_REC_SIZE, OLD_REC_SIZE, DIFFERENCE ; 0406
                                           0B  19 00037                BLSS    2$                  ; 0418
                3E        00A2 CA            02  E1 00039                BBC     #2, 162(IFAB), 4$   ; 0420
                               01            52  D1 0003F                CMPL    DIFFERENCE, #1      ; 0422
                                             39  12 00042                BNEQ    4$
                0A        00A2 CA            02  E1 00044 2$:           BBC     #2, 162(IFAB), 3$   ; 0426
```

```
                                01            52  D1 0004A          CMPL    DIFFERENCE, #1                                  ; 0428
                                              05  12 0004D          BNEQ    3$
                             07 A9        80  8F  88 0004F          BICB2   #128, 7(IRAB)                                   ; 0430
                          50    51            53  C1 00054 3$:      ADDL3   OLD_REC_SIZE, OLD_REC_ADDR, R0                  ; 0432
                          52    50            56  C3 00058          SUBL3   REC_ADDR, R0, DIFFERENCE
                                51            56  D0 0005C          MOVL    REC_ADDR, OLD_REC_ADDR                          ; 0433
                                56        58  A9  D0 0005F          MOVL    88(IRAB), REC_ADDR                              ; 0437
                                5C            03  D0 00063          MOVL    #3, AP                                          ; 0439
                                50      00B4  CA  3C 00066          MOVZWL  180(IFAB), R0                                   ; 0440
                                    60  B940  3F  0006B          PUSHAW  @96(IRAB)[R0]
                                      0000G 30  0006F          BSBW    RM$RECORD_KEY
                                5E            04  C0 00072          ADDL2   #4, SP
                                          04  AE  D4 00075          CLRL    STATUS                                          ; 0442
                                56            51  D0 00078          MOVL    OLD_REC_ADDR, REC_ADDR                          ; 0443
                                              55  11 0007B          BRB     11$                                             ; 0418
                                66        40  8F  8A 0007D 4$:      BICB2   #64, (REC_ADDR)                                 ; 0459
                                              52  D5 00081          TSTL    DIFFERENCE                                      ; 0463
                                              2A  13 00083          BEQL    8$
                          05    00A2        CA  02 E0 00085          BBS     #2, 162(IFAB), 5$                               ; 0472
                                0A    00A1  CA  E9 0008B          BLBC    161(IFAB), 6$                                   ; 0474
                                0C    00A1  CA  E9 00090 5$:      BLBC    161(IFAB), 7$                                   ; 0476
                                01            52  D1 00095          CMPL    DIFFERENCE, #1                                  ; 0478
                                              07  12 00098          BNEQ    7$
                                FE  A1        20  AE  B0 0009A 6$:      MOVW    NEW_REC_SIZE, -2(OLD_REC_ADDR)                  ; 0480
                                              0E  11 0009F          BRB     8$
                                66        40  8F  88 000A1 7$:      BISB2   #64, (REC_ADDR)                                 ; 0495
                                              52  D4 000A5          CLRL    DIFFERENCE                                      ; 0496
                                        FE A341  9F  000A7          PUSHAB  -2(OLD_REC_SIZE)[OLD_REC_ADDR]                  ; 0498
                                9E        24  AE  B0 000AB          MOVW    NEW_REC_SIZE, @(SP)+
                                03    00B7  CA  91 000AF 8$:      CMPB    183(IFAB), #3                                   ; 0504
                                              06  1E 000B4          BGEQU   9$
                                50        58  A9  D0 000B6          MOVL    88(IRAB), BUFFER                                ; 0506
                                              04  11 000BA          BRB     10$
                                50        68  A9  D0 000BC 9$:      MOVL    104(IRAB), BUFFER                               ; 0508
                                              03  BB 000C0 10$:     PUSHR   #^M<R0,R1>                                      ; 0510
                                          28  AE  DD 000C2          PUSHL   NEW_REC_SIZE
                                        0000G 30  000C5          BSBW    RM$MOVE
                                5E            0C  C0 000C8          ADDL2   #12, SP
                                51            50  D0 000CB          MOVL    R0, OLD_REC_ADDR
                             04 AE            01  D0 000CE          MOVL    #1, STATUS                                      ; 0512
                                              52  D5 000D2 11$:     TSTL    DIFFERENCE                                      ; 0525
                                              1E  13 000D4          BEQL    12$
                             04 A5            52  A2 000D6          SUBW2   DIFFERENCE, 4(BKT_ADDR)                         ; 0530
                          50    55            51  C3 000DA          SUBL3   OLD_REC_ADDR, BKT_ADDR, R0                      ; 0533
                                53        04  A5  3C 000DE          MOVZWL  4(BKT_ADDR), LENGTH
                                53            50  C0 000E2          ADDL2   R0, LENGTH
                                              0D  15 000E5          BLEQ    12$                                             ; 0535
                                              51  DD 000E7          PUSHL   OLD_REC_ADDR                                    ; 0537
                                            6241  9F 000E9          PUSHAB  (DIFFERENCE)[OLD_REC_ADDR]
                                              53  DD 000EC          PUSHL   LENGTH
                                        0000G 30  000EE          BSBW    RM$MOVE
                                5E            0C  C0 000F1          ADDL2   #12, SP
                          1E    1C  A7          06  E1 000F4 12$:     BBC     #6, 28(IDX_DFN), 13$                            ; 0546
                                          04  AE  D5 000F9          TSTL    STATUS                                          ; 0548
                                              19  12 000FC          BNEQ    13$
                                              53  D5 000FE          TSTL    LENGTH                                          ; 0550
                                              15  15 00100          BLEQ    13$
```

```
                 11        66          03  E0 00102          BBS     #3, (REC_ADDR), 13$                    ; 0552
                 51        56          6E  C1 00106          ADDL3   RECORD_OVHD, REC_ADDR, R1              ; 0554
                           50    00B4  CA  3C 0010A          MOVZWL  180(IFAB), R0                          ;
                           50      60  B940  DE 0010F        MOVAL   @96(IRAB)[R0], R0                      ;
                                       0000G 30 00114        BSBW    RM$EXPAND_KEYD                         ;
                           50      04  AE  D0 00117 13$:     MOVL    STATUS, R0                             ; 0558
                           5E          08  C0 0011B          ADDL2   #8, SP                                 ; 0560
                                   091C  8F  BA 0011E        POPR    #^M<R2,R3,R4,R8,R11>                   ;
                                             05 00122        RSB                                            ;

; Routine Size:  291 bytes,    Routine Base:  RM$RMS3 + 0000

;  498          0561  1
```

```
;  500     0562  1 %SBTTL  'RM$UPDATE_SCAN'
;  501     0563  1 ROUTINE RM$UPDATE_SCAN : RL$UPDATE_SCAN =
;  502     0564  1
;  503     0565  1 !++
;  504     0566  1 !
;  505     0567  1 ! FUNCTIONAL DESCRIPTION:
;  506     0568  1 !
;  507     0569  1 !       Compare the old and new records to determine which new keys are to be
;  508     0570  1 !       inserted and which old keys are to be deleted. If the file is a prologue
;  509     0571  1 !       3 file, then unpack the primary data record into the old record buffer,
;  510     0572  1 !       and if there are old alternate keys to be deleted and the file is a
;  511     0573  1 !       prologue 1 or 2 file, move the primary data record into the old record
;  512     0574  1 !       buffer as well.
;  513     0575  1 !
;  514     0576  1 ! CALLING SEQUENCE:
;  515     0577  1 !
;  516     0578  1 !       RM$UPDATE_SCAN()
;  517     0579  1 !
;  518     0580  1 ! INPUT PARAMETERS:
;  519     0581  1 !       NONE
;  520     0582  1 !
;  521     0583  1 ! IMPLICIT INPUTS:
;  522     0584  1 !
;  523     0585  1 !       IDX_DFN                 - address of the primary key index descriptor
;  524     0586  1 !           IDX$V_CHGKEYS       - if set, keys are allowed to change
;  525     0587  1 !           IDX$B_DESC_NO       - descriptor number
;  526     0588  1 !           IDX$B_KEYSZ         - size of key
;  527     0589  1 !           IDX$W_MINRECSZ      - minimum size of record containing this key
;  528     0590  1 !
;  529     0591  1 !       IFAB                    - address of IFAB
;  530     0592  1 !           IFB$B_PLG_VER       - prologue version of file
;  531     0593  1 !
;  532     0594  1 !       IRAB                    - address of IRAB
;  533     0595  1 !           IRB$B_CUR_KREF      - current NRP key of reference
;  534     0596  1 !           IRB$L_OLDBUF        - address of old record buffer
;  535     0597  1 !           IRB$L_RBF           - address of new record buffer
;  536     0598  1 !           IRB$W_RSZ           - size of new record
;  537     0599  1 !           IRB$V_UPDATE_IF     - if set, $PUT converted into an $UPDATE
;  538     0600  1 !           IRB$L_UPDBUF        - address of update buffer
;  539     0601  1 !
;  540     0602  1 !       REC_ADDR                - address of primary data record
;  541     0603  1 !
;  542     0604  1 ! OUTPUT PARAMETERS:
;  543     0605  1 !       NONE
;  544     0606  1 !
;  545     0607  1 ! IMPLICIT OUTPUTS:
;  546     0608  1 !
;  547     0609  1 !       IRAB                    - address of IRAB
;  548     0610  1 !           IRB$V_FIND_LAST     - set to 0
;  549     0611  1 !
;  550     0612  1 ! ROUTINE VALUE:
;  551     0613  1 !
;  552     0614  1 !       CHG - if primary key changes key value at all, or any alternate key
;  553     0615  1 !             changes values and CHGKEYS not set
;  554     0616  1 !       SUC - otherwise
;  555     0617  1 !
;  556     0618  1 ! SIDE EFFECTS:
```

```
;  557     0619  1 !
;  558     0620  1 !       AP is trashed.
;  559     0621  1 !       Each byte in the update buffer corresponds to an index with:
;  560     0622  1 !           INS_NEW set if a new key value is to be inserted
;  561     0623  1 !           OLD_DEL set if the old key value requires deletion.
;  562     0624  1 !       The same flags for the primary key byte indicate whether any
;  563     0625  1 !           alternate keys require insertion or deletion at all.
;  564     0626  1 !       If the file is a prologue 3 file then the old record has been unpacked
;  565     0627  1 !           into the old record buffer.
;  566     0628  1 !       If the file is a prologue 1 or 2 file, and there are old altenate keys
;  567     0629  1 !           to be deleted then the old record has been moved into the old record
;  568     0630  1 !           buffer.
;  569     0631  1 !
;  570     0632  1 !--
;  571     0633  1
;  572     0634  2     BEGIN
;  573     0635  2
;  574     0636  2     EXTERNAL REGISTER
;  575     0637  2         R_IDX_DFN_STR,
;  576     0638  2         R_IFAB_STR,
;  577     0639  2         R_IMPURE,
;  578     0640  2         R_IRAB_STR,
;  579     0641  2         R_RAB,
;  580     0642  2         R_REC_ADDR_STR;
;  581     0643  2
;  582     0644  2     BUILTIN
;  583     0645  2         AP;
;  584     0646  2
;  585     0647  2     LOCAL
;  586     0648  2         OLD_REC_ADDR,
;  587     0649  2         OLD_REC_SIZE    : WORD;
;  588     0650  2
;  589     0651  2     ! Retrieve the size of the old record, as it is in the bucket, and set
;  590     0652  2     ! up a pointer to the old data record itself.
;  591     0653  2     !
;  592     0654  3     BEGIN
;  593     0655  3
;  594     0656  3     LOCAL
;  595     0657  3         REC_SIZE;
;  596     0658  3
;  597     0659  3     OLD_REC_ADDR = .REC_ADDR + RM$REC_OVHD(0; REC_SIZE);
;  598     0660  3     OLD_REC_SIZE = .REC_SIZE;
;  599     0661  2     END;
;  600     0662  2
;  601     0663  2     ! If the file is a prologue 3 file, then the old record must be unpacked
;  602     0664  2     ! into the old record buffer. Reset the pointer to the data portion of the
;  603     0665  2     ! old record so that it points into the old record buffer.
;  604     0666  2     !
;  605     0667  2     IF .IFAB[IFB$B_PLG_VER] EQLU PLG$C_VER_3
;  606     0668  2     THEN
;  607     0669  3         BEGIN
;  608     0670  3
;  609     0671  3         GLOBAL REGISTER
;  610     0672  3             R_BKT_ADDR;
;  611     0673  3
;  612     0674  3         LOCAL
;  613     0675  3             SAVE_REC_ADDR : REF BBLOCK;
```

```
;  614          0676  3
;  615          0677  3          SAVE_REC_ADDR = .REC_ADDR;
;  616          0678  3          REC_ADDR = .OLD_REC_ADDR;
;  617          0679  3          OLD_REC_ADDR = .IRAB[IRB$L_OLDBUF] + 2;
;  618          0680  3
;  619          0681  3          ! If this record is marked RU_UPDATE then the last two bytes of the
;  620          0682  3          ! reserved space of the record contain the actual size of the record.
;  621          0683  3          !
;  622          0684  3          IF .SAVE_REC_ADDR[IRC$V_RU_UPDATE]
;  623          0685  3          THEN
;  624          0686  3              OLD_REC_SIZE = .(.REC_ADDR + .OLD_REC_SIZE - IRC$C_DATSZFLD)<0,16>;
;  625          0687  3
;  626          0688  3          ! If this operation is a $PUT converted into an $UPDATE, then the
;  627          0689  3          ! primary key of the old record maybe found in keybuffer 2. Likewise,
;  628          0690  3          ! if this is a $UPDATE and RMS positioned to the record by means of the
;  629          0691  3          ! primary key of reference although not by a $FIND, then the primary
;  630          0692  3          ! key of the old record maybe found in keybuffer 1. Otherwise, a bucket
;  631          0693  3          ! scan must be done to re-expand the primary key.
;  632          0694  3          !
;  633          0695  3          IF .IRAB[IRB$V_UPDATE_IF]
;  634          0696  3          THEN
;  635          0697  3              AP = 2
;  636          0698  3          ELSE
;  637          0699  3              IF  .IRAB[IRB$B_CUR_KREF] EQLU 0
;  638          0700  3                  AND
;  639          0701  3                  NOT .IRAB[IRB$V_FIND_LAST]
;  640          0702  3              THEN
;  641          0703  3                  AP = 1
;  642          0704  3              ELSE
;  643          0705  3                  AP = 0;
;  644          0706  3
;  645          0707  3              OLD_REC_SIZE = RM$UNPACK_REC (.OLD_REC_ADDR, .OLD_REC_SIZE);
;  646          0708  3              (.IRAB[IRB$L_OLDBUF])<0,16> = .OLD_REC_SIZE;
;  647          0709  3              REC_ADDR = .SAVE_REC_ADDR
;  648          0710  2              END;
;  649          0711  2
;  650          0712  2          IRAB[IRB$V_FIND_LAST] = 0;
;  651          0713  2
;  652          0714  2          ! If the primary key has changed, return an error. This comparison does
;  653          0715  2          ! not have to be done if the operation is an UPDATE_IF.
;  654          0716  2          !
;  655          0717  2          IF NOT .IRAB[IRB$V_UPDATE_IF]
;  656          0718  2          THEN
;  657          0719  3              BEGIN
;  658          0720  3              AP = 0;
;  659          0721  3
;  660          0722  3              IF RM$COMPARE_KEY (.OLD_REC_ADDR,
;  661          0723  3                                 .IRAB[IRB$L_RBF],
;  662          0724  3                                 .IDX_DFN[IDX$B_KEYSZ])
;  663          0725  3              THEN
;  664          0726  3                  RETURN RMSERR (CHG);
;  665          0727  2              END;
;  666          0728  2
;  667          0729  2          ! The INS_NEW and OLD_DEL flags for the first byte of the update buffer
;  668          0730  2          ! (which would be for key 0) are used to indicate whether any inserts or
;  669          0731  2          ! deletes are to be done at all. Zero these.
;  670          0732  2          !
```

```
;  671     0733  2      BBLOCK[.IRAB[:RB$L_UPDBUF], UPD$B_FLAGS] = 0;
;  672     0734  2
;  673     0735  2      ! Process all of the alternate keys determing whether the alternate key has
;  674     0736  2      ! changed, the old key must be deleted, and a new key must be inserted.
;  675     0737  2      !
;  676     0738  2      WHILE RM$GET_NEXT_KEY()
;  677     0739  2      DO
;  678     0740  3          BEGIN
;  679     0741  3
;  680     0742  3          LOCAL
;  681     0743  3              UPD_BUF_ADDR        : REF BBLOCK;
;  682     0744  3
;  683     0745  3          ! Point to the update buffer associated with this index.
;  684     0746  3          !
;  685     0747  3          UPD_BUF_ADDR = .IRAB[IRB$L_UPDBUF] + .IDX_DFN[IDX$B_DESC_NO];
;  686     0748  3          UPD_BUF_ADDR[UPD$B_FLAGS] = 0;
;  687     0749  3
;  688     0750  3          AP = 0;
;  689     0751  3
;  690     0752  3          ! Set the flag INS_NEW for this key of reference if the key is present
;  691     0753  3          ! in the new record, disregarding for the time being whether the key
;  692     0754  3          ! is present in the old record and whether the two keys are the same.
;  693     0755  3          !
;  694     0756  3          IF .IRAB[IRB$W_RSZ] GEQU .IDX_DFN[IDX$W_MINRECSZ]
;  695     0757  3              AND
;  696     0758  3             RM$NULLKEY (.IRAB[IRB$L_RBF])
;  697     0759  3          THEN
;  698     0760  3              UPD_BUF_ADDR[UPD$V_INS_NEW] = 1;
;  699     0761  3
;  700     07o2  3          ! Determine whether this same key was present in the old record.
;  701     0763  3          !
;  702     0764  3          IF .OLD_REC_SIZE GEQU .IDX_DFN[IDX$W_MINRECSZ]
;  703     0765  3              AND
;  704     0766  3             RM$NULLKEY (.OLD_REC_ADDR)
;  705     0767  3          THEN
;  706     0768  3
;  707     0769  3              ! If the key was present in the old record but is not present in
;  708     0770  3              ! the new record then flag that the old key for this key of
;  709     0771  3              ! reference should be deleted.
;  710     0772  3              !
;  711     0773  3              IF NOT .UPD_BUF_ADDR[UPD$V_INS_NEW]
;  712     0774  3              THEN
;  713     0775  3                  UPD_BUF_ADDR[UPD$V_OLD_DEL] = 1
;  714     0776  3
;  715     0777  3              ! If the key is present in both records then compare the two keys.
;  716     0778  3              !
;  717     0779  3              ELSE
;  718     0780  4                  BEGIN
;  719     0781  4                  AP = 0;
;  720     0782  4
;  721     0783  4                  ! If there are keys present in both records but they are
;  722     0784  4                  ! different then flag that there is a new key to be inserted
;  723     0785  4                  ! (already has been done) and an old key to be deleted for
;  724     0786  4                  ! this key of reference.
;  725     0787  4                  !
;  726     0788  4                  IF RM$COMPARE_KEY (.OLD_REC_ADDR,
;  727     0789  4                                     .IRAB[IRB$L_RBF],
```

```
;  728     0790  4                                   .IDX_DFN[IDX$B_KEYSZ])
;  729     0791  4                THEN
;  730     0792  4                    UPD_BUF_ADDR[UPD$V_OLD_DEL] = 1
;  731     0793  4
;  732     0794  4                ! If there are keys present in both records and they are
;  733     0795  4                ! identical, then there is no need to insert or delete any
;  734     0796  4                ! keys for this key of reference.
;  735     0797  4                !
;  736     0798  4                ELSE
;  737     0799  4                    UPD_BUF_ADDR[UPD$V_INS_NEW] = 0;
;  738     0800  3                END;
;  739     0801  3
;  740     0802  3            ! If the keys have changed between the two records, and this key of
;  741     0803  3            ! reference does not allow keys to change, return an error.
;  742     0804  3            !
;  743     0805  3            IF .UPD_BUF_ADDR[UPD$B_FLAGS] NEQU 0
;  744     0806  3                AND
;  745     0807  3               NOT .IDX_DFN[IDX$V_CHGKEYS]
;  746     0808  3            THEN
;  747     0809  3                RETURN RMSERR (CHG);
;  748     0810  3
;  749     0811  3            ! Summarize the key 0 flags to this point in the key-by-key comparison.
;  750     0812  3            !
;  751     0813  3            BBLOCK[.IRAB[IRB$L_UPDBUF], UPD$B_FLAGS] =
;  752     0814  3                            .BBLOCK[.IRAB[IRB$L_UPDBUF], UPD$B_FLAGS]
;  753     0815  3                             OR
;  754     0816  3                            .UPD_BUF_ADDR[UPD$B_FLAGS]
;  755     0817  3
;  756     0818  2            END;
;  757     0819  2
;  758     0820  2            ! If there are any old keys which will have to be deleted, and this is a
;  759     0821  2            ! prologue 1 or 2 file, then save the old record in the old record buffer.
;  760     0822  2            !
;  761     0823  2            IF  .BBLOCK[.IRAB[IRB$L_UPDBUF], UPD$V_OLD_DEL]
;  762     0824  2                 AND
;  763     0825  2                .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  764     0826  2            THEN
;  765     0827  3                BEGIN
;  766     0828  3
;  767     0829  3                GLOBAL REGISTER
;  768     0830  3                    R_BDB;
;  769     0831  3
;  770     0832  3                RM$MOVE (.OLD_REC_SIZE, .OLD_REC_ADDR, .IRAB[IRB$L_OLDBUF] + 2);
;  771     0833  3                (.IRAB[IRB$L_OLDBUF])<0,16> = .OLD_REC_SIZE;
;  772     0834  2                END;
;  773     0835  2
;  774     0836  2            ! Reset to primary key descriptor and return success.
;  775     0837  2            !
;  776     0838  2            RETURN RM$KEY_DESC(0)
;  777     0839  2
;  778     0840  1            END;
```

```
                                                                 PUSHR   #^M<R2,R3,R4,R5>                          ; 0563
                                                                 CLRL    R1                                       ; 0659
                                                                 BSBW    RMS$REC_OVHD                             ;
                                                                 ADDL3   REC_ADDR, R0, OLD_REC_ADDR               ;
                                                                 MOVW    REC_SIZE, OLD_REC_SIZE                   ; 0660
                                                                 CMPB    183(IFAB), #3                            ; 0667
                                                                 BNEQ    5$                                       ;
                                                                 MOVL    REC_ADDR, SAVE_REC_ADDR                  ; 0677
                                                                 MOVL    OLD_REC_ADDR, REC_ADDR                   ; 0678
                                                                 ADDL3   #2, 108(IRAB), OLD_REC_ADDR              ; 0679
                                                                 BBC     #6, (SAVE_REC_ADDR), 1$                  ; 0684
                                                                 MOVZWL  OLD_REC_SIZE, R0                         ; 0686
                                                                 PUSHAB  -2(R0)[REC_ADDR]                         ;
                                                                 MOVW    @(SP)+, OLD_REC_SIZE                     ;
                                                         1$:     BBC     #4, 6(IRAB), 2$                          ; 0695
                                                                 MOVL    #2, AP                                   ; 0697
                                                                 BRB     4$                                       ;
                                                         2$:     TSTB    195(IRAB)                                ; 0699
                                                                 BNEQ    3$                                       ;
                                                                 BBS     #5, 4(IRAB), 3$                          ; 0701
                                                                 MOVL    #1, AP                                   ; 0703
                                                                 BRB     4$                                       ;
                                                         3$:     CLRL    AP                                       ; 0705
                                                         4$:     MOVZWL  OLD_REC_SIZE, R1                         ; 0707
                                                                 MOVL    OLD_REC_ADDR, R0                         ;
                                                                 BSBW    RMS$UNPACK_REC                           ;
                                                                 MOVW    R0, OLD_REC_SIZE                         ;
                                                                 MOVW    OLD_REC_SIZE, @108(IRAB)                 ; 0708
                                                                 MOVL    SAVE_REC_ADDR, REC_ADDR                  ; 0709
                                                                 BICB2   #32, 4(IRAB)                             ; 0712
                                                                 BBS     #4, 6(IRAB), 6$                          ; 0717
                                                                 CLRL    AP                                       ; 0720
                                                                 MOVZBL  32(IDX_DFN), R0                          ; 0722
                                                                 MOVL    88(IRAB), R3                             ;
                                                                 MOVL    OLD_REC_ADDR, R1                         ;
                                                                 BSBW    RMS$COMPARE_KEY                          ;
                                                                 BLBS    R0, 12$                                  ;
                                                         6$:     CLRB    @100(IRAB)                               ; 0733
                                                         7$:     BSBW    RMS$GET_NEXT_KEY                         ; 0738
                                                                 BLBC    R0, 14$                                  ;
                                                                 MOVZBL  16(IDX_DFN), UPD_BUF_ADDR                ; 0747
                                                                 ADDL2   100(IRAB), UPD_BUF_ADDR                  ;
                                                                 CLRB    (UPD_BUF_ADDR)                           ; 0748
                                                                 CLRL    AP                                       ; 0750
                                                                 CMPW    86(IRAB), 34(IDX_DFN)                    ; 0756
                                                                 BLSSU   8$                                       ;
                                                                 PUSHL   88(IRAB)                                 ; 0758
                                                                 BSBW    RMS$NULLKEY                              ;
                                                                 ADDL2   #4, SP                                   ;
                                                                 BLBC    R0, 8$                                   ;
                                                                 BISB2   #1, (UPD_BUF_ADDR)                       ; 0760
                                                         8$:     CMPW    OLD_REC_SIZE, 34(IDX_DFN)                ; 0764
                                                                 BLSSU   11$                                      ;
                                                                 PUSHL   OLD_REC_ADDR                             ; 0766
                                                                 BSBW    RMS$NULLKEY                              ;
                                                                 ADDL2   #4, SP                                   ;
                                                                 BLBC    R0, 11$                                  ;
```

```
                                 13          65  E9 000B5          BLBC    (UPD_BUF_ADDR), 9$                      ; 0773
                                             5C  D4 000B8          CLRL    AP                                      ; 0781
                                 50      20  A7  9A 000BA          MOVZBL  32(IDX_DFN), R0                         ; 0788
                                 53      58  A9  D0 000BE          MOVL    88(IRAB), R3                            ;
                                 51          54  D0 000C2          MOVL    OLD_REC_ADDR, R1                        ;
                                         0000G 30 000C5          BSBW    RM$COMPARE_KEY                          ;
                                 05          50  E9 000C8          BLBC    R0, 10$                                 ;
                                 65          02  88 000CB 9$:      BISB2   #2, (UPD_BUF_ADDR)                      ; 0792
                                             03  11 000CE          BRB     11$                                     ;
                                 65          01  8A 000D0 10$:     BICB2   #1, (UPD_BUF_ADDR)                      ; 0799
                                             65  95 000D3 11$:     TSTB    (UPD_BUF_ADDR)                          ; 0805
                                             0C  13 000D5          BEQL    13$                                     ;
                     07      1C  A7              01  E0 000D7          BBS     #1, 28(IDX_DFN), 13$                    ; 0807
                                 50      849C    8F  3C 000DC 12$:     MOVZWL  #33948, R0                      ; 0809
                                             2E  11 000E1          BRB     16$                                     ;
                             64  B9              65  88 000E3 13$:     BISB2   (UPD_BUF_ADDR), @100(IRAB)      ; 0816
                                             93  11 000E7          BRB     7$                                      ; 0813
                     1B      64  B9              01  E1 000E9 14$:     BBC     #1, @100(IRAB), 15$             ; 0823
                                 03      00B7    CA  91 000EE          CMPB    183(IFAB), #3                   ; 0825
                                             14  1E 000F3          BGEQU   15$                                     ;
                     7E      6C  A9              02  C1 000F5          ADDL3   #2, 108(IRAB), -(SP)            ; 0832
                                             54  DD 000FA          PUSHL   OLD_REC_ADDR                            ;
                                 7E          52  3C 000FC          MOVZWL  OLD_REC_SIZE, -(SP)                     ;
                                         0000G 30 000FF          BSBW    RM$MOVE                                 ;
                                 5E          0C  C0 00102          ADDL2   #12, SP                                 ;
                             6C  B9              52  B0 00105          MOVW    OLD_REC_SIZE, @108(IRAB)        ; 0833
                                             7E  D4 00109 15$:     CLRL    -(SP)                           ; 0838
                                         0000G 30 0010B          BSBW    RM$KEY_DESC                             ;
                                 5E          04  C0 0010E          ADDL2   #4, SP                                  ;
                                             3C  BA 00111 16$:     POPR    #^M<R2,R3,R4,R5>                        ; 0840
                                                 05 00113          RSB                                             ;

; Routine Size:  276 bytes,    Routine Base:  RM$RMS3 + 0123

;  779          0841  1
```

```
;  781          0842  1 %SBTTL  'RM$UPDATE3B'
;  782          0843  1 GLOBAL ROUTINE RM$UPDATE3B : RL$RABREG_67 =
;  783          0844  1
;  784          0845  1 !++
;  785          0846  1 !
;  786          0847  1 ! FUNCTIONAL DESCRIPTION:
;  787          0848  1 !
;  788          0849  1 !       High level update routine, using the following update algorithm:
;  789          0850  1 !
;  790          0851  1 !       1. If the operation is an $UPDATE (and not a converted $PUT), then the
;  791          0852  1 !          current primary data record is retrieved using its RFA saved as part
;  792          0853  1 !          of the next record context.
;  793          0854  1 !
;  794          0855  1 !       2. Record sizes, null key status, and comparisons between the new and
;  795          0856  1 !          old versions of the current record are used to set flags in the
;  796          0857  1 !          update buffer to determine which new secondary key values require
;  797          0858  1 !          insertion and which old secondary key values require deletion. At
;  798          0859  1 !          same time illegal key changes are detected.
;  799          0860  1 !
;  800          0861  1 !       3. If the file is a prologue 3 file, then the old version of the current
;  801          0862  1 !          primary data record is saved in expanded format within the old record
;  802          0863  1 !          buffer. If the file is a prologue 1 or 2 file, and there are old
;  803          0864  1 !          secondary keys to be deleted, then the old version of the current
;  804          0865  1 !          record will also be moved into the old record buffer.
;  805          0866  1 !
;  806          0867  1 !       4. New secondary key values are inserted. This involves:
;  807          0868  1 !               a. Releasing the primary data bucket containing the current
;  808          0869  1 !                  primary data record.
;  809          0870  1 !               b. Inserting the new secondary keys requiring insertion.
;  810          0871  1 !               c. Reclaiming the primary data bucket containing the current
;  811          0872  1 !                  primary data record.
;  812          0873  1 !
;  813          0874  1 !          If the file is being RU ROLLBACK Recovered, then as the new SIDRs
;  814          0875  1 !          already exists in the file they are un-deleted instead of inserted.
;  815          0876  1 !
;  816          0877  1 !       5. The new version of the current record is inserted in place of the old
;  817          0878  1 !          version, and any primary data bucket splits that are required in
;  818          0879  1 !          order to make room for the new record are performed as is all
;  819          0880  1 !          required updating of the primary index.
;  820          0881  1 !
;  821          0882  1 !          If the file is being RU Journalled and a Recovery Unit is in
;  822          0883  1 !          progress, then the record is not allowed to shrink in size, and if
;  823          0884  1 !          necessary even increases in size by one byte to reserve the space
;  824          0885  1 !          the record currently occupies in case the Recovery Unit is aborted.
;  825          0886  1 !
;  826          0887  1 !       6. Old secondary key values requiring deletion are deleted. The keys
;  827          0888  1 !          themselves are extracted from the old version of the current primary
;  828          0889  1 !          data record which was saved earlier in the old record buffer.
;  829          0890  1 !
;  830          0891  1 !          If the file is being RU Journalled and a Recovery Unit is in
;  831          0892  1 !          progress, then the old SIDRs are just marked RU_DELETE and no space
;  832          0893  1 !          is reclaimed.
;  833          0894  1 !
;  834          0895  1 ! CALLING SEQUENCE:
;  835          0896  1 !
;  836          0897  1 !       BSBW RM$UPDATE3B()
;  837          0898  1 !
```

```
;  838     0899  1 ! INPUT PARAMETERS:
;  839     0900  1 !       NONE
;  840     0901  1 !
;  841     0902  1 ! IMPLICIT INPUTS:
;  842     0903  1 !
;  843     0904  1 !       IDX_DFN                         - primary key index descriptor
;  844     0905  1 !           IDX$B_DATBKTYP              - primary data bucket type
;  845     0906  1 !           IDX$B_KEYSZ                 - size of primary key
;  846     0907  1 !
;  847     0908  1 !       IFAB                            - address of IFAB
;  848     0909  1 !           IFB$B_PLG_VER               - prologue version of file
;  849     0910  1 !           IFB$B_RFMORG                - format of records
;  850     0911  1 !           IFB$V_RUP                   - if set, Recovery Unit in progress
;  851     0912  1 !           IFB$V_RU_RECVR              - if set, RU ROLLBACK in progress
;  852     0913  1 !
;  853     0914  1 !       IRAB                            - address of IRAB
;  854     0915  1 !           IRB$V_ABOVELCKD             - if set, above level bucket is locked
;  855     0916  1 !           IRB$L_CURBDB                - address of primary data bucket's BDB
;  856     0917  1 !           IRB$B_CUR_KREF              - current NRP key of reference
;  857     0918  1 !           IRB$L_LOCK_BDB              - address of above level bucket's BDB
;  858     0919  1 !           IRB$L_OLDBUF                - address of old record buffer
;  859     0920  1 !           IRB$W_PUTUP_ID              - ID of current primary data record
;  860     0921  1 !           IRB$L_PUTUPD_VBN            - VBN of current primary data record
;  861     0922  1 !           IRB$W_RSZ                   - size of updated record
;  862     0923  1 !           IRB$V_UPDATE                - if set, $UPDATE requested
;  863     0924  1 !           IRB$V_UPDATE_IF             - if set, converted $PUT requested
;  864     0925  1 !           IRB$L_UPDBUF                - address of update buffer
;  865     0926  1 !
;  866     0927  1 !       REC_ADDR                        - address of current primary data record
;  867     0928  1 !
;  868     0929  1 ! OUTPUT PARAMETERS:
;  869     0930  1 !       NONE
;  870     0931  1 !
;  871     0932  1 ! IMPLICIT OUTPUTS:
;  872     0933  1 !
;  873     0934  1 !       IRAB                            - address of IRAB
;  874     0935  1 !           IRB$B_CACHEFLGS             - CSH$V_LOCK maybe set
;  875     0936  1 !           IRB$V_IDX_ERR               - if set, error updating index
;  876     0937  1 !           IRB$W_LAST_ID               - ID of record's last known position
;  877     0938  1 !           IRB$L_LAST_VBN              - VBN of record's last known position
;  878     0939  1 !           IRB$L_LST_NCMP              - last zero-front compressed key record
;  879     0940  1 !           IRB$W_SRCHFLAGS             - IRB$V_POSINSERT maybe set
;  880     0941  1 !           IRB$B_STOPLEVEL             - maybe set either to 0 or 1
;  881     0942  1 !
;  882     0943  1 ! ROUTINE VALUE:
;  883     0944  1 !
;  884     0945  1 !       See errors from:
;  885     0946  1 !
;  886     0947  1 !       RM$FIND_BY_RFA
;  887     0948  1 !       RM$FIND_BY_RRV
;  888     0949  1 !       RM$INS_ALL_SIDR
;  889     0950  1 !       RM$INSS_OR_IDX
;  890     0951  1 !       RM$PUT_UPD_CHKS
;  891     0952  1 !       RM$PUT_UPD_FIN
;  892     0953  1 !       RM$PUT_UPD_SPL
;  893     0954  1 !       RM$RLSBKT
;  894     0955  1 !       RM$RU_JOURNAL3
```

```
;  895          0956  1 !       RM$UPDATE_SCAN
;  896          0957  1 !       RM$UPDELCOM
;  897          0958  1 !
;  898          0959  1 ! SIDE EFFECTS:
;  899          0960  1 !       NONE
;  900          0961  1 !
;  901          0962  1 !--
;  902          0963  1
;  903          0964  2     BEGIN
;  904          0965  2
;  905          0966  2     BUILTIN
;  906          0967  2         AP,
;  907          0968  2         TESTBITSS,
;  908          0969  2         TESTBITCS;
;  909          0970  2
;  910          0971  2     LABEL
;  911          0972  2         GETBACK;
;  912          0973  2
;  913          0974  2     EXTERNAL REGISTER
;  914          0975  2         COMMON_RAB_STR,
;  915          0976  2         R_IDX_DFN_STR,
;  916          0977  2         R_REC_ADDR_STR;
;  917          0978  2
;  918          0979  2     GLOBAL REGISTER
;  919          0980  2         COMMON_IO_STR;
;  920          0981  2
;  921          0982  2     ! This routine maybe called as the result of a $PUT converted into an
;  922          0983  2     ! $UPDATE, or as the result of an $UPDATE. In the former case, this routine
;  923          0984  2     ! would have been called with the primary data bucket locked, and RMS
;  924          0985  2     ! positioned to the record to be updated so there would be no need to
;  925          0986  2     ! perform this positioning. In the case of a straight $UPDATE, it would be
;  926          0987  2     ! necessary to perform this positioning. To differentiate between th  two
;  927          0988  2     ! cases, the IRAB bit IRB$V_UPDATE is set on $UPDATEs but cleared on
;  928          0989  2     ! converted $PUTs on entry to this routine.
;  929          0990  2     !
;  930          0991  2     ! On any errors from this point on, the routine RM$CLEAN_BDB will be called
;  931          0992  2     ! to make sure any locked buckets, including the bucket which might have
;  932          0993  2     ! been locked during lockabove optimization, are realeased.
;  933          0994  2     !
;  934          0995  2     IF TESTBITSS (IRAB[IRB$V_UPDATE])
;  935          0996  2     THEN
;  936          0997  3         BEGIN
;  937          0998  3
;  938          0999  3         ! This is an $UPDATE so make sure IRB$V_ABOVELCKD is clear as there
;  939          1000  3         ! cannot be any bucket locked at this time. Position to the record to
;  940          1001  3         ! be updated.
;  941          1002  3         !
;  942          1003  3         IR     RB$V_ABOVELCKD] = 0;
;  943          1004  3
;  944          1005  3         RETURN_ON_ERROR (RM$UPDDELCOM());
;  945          1006  2         END;
;  946          1007  2
;  947          1008  2     ! Perform some common checks and set up the primary key descriptor. These
;  948          1009  2     ! checks have already been performed if this is a RU ROLLBACK Recovery
;  949          1010  2     ! operation.
;  950          1011  2     !
;  951          1012  2     IF NOT .IFAB[IFB$V_RU_RECVR]
```

```
;  952          1013  2     THEN
;  953          1014  2         RETURN_ON_ERROR (RM$PUT_UPD_CHKS(), RM$CLEAN_BDB());
;  954          1015  2
;  955          1016  2     ! Set up the update buffer indicating which alternate keys in the record to
;  956          1017  2     ! be updated will need inserting and which will need deleting.
;  957          1018  2     !
;  958          1019  2     RETURN_ON_ERROR (RM$UPDATE_SCAN(), RM$CLEAN_BDB());
;  959          1020  2
;  960          1021  2     IRAB[IRB$W_LAST_ID] = IRC$_ID(REC_ADDR);
;  961          1022  2
;  962          1023  2     ! Before any permanent modification to the file takes place, RU Journal this
;  963          1024  2     ! operation if RU Journalling is enabled and the process is within a
;  964          1025  2     ! Recovery Unit.
;  965          1026  2     !
;  966          1027  2     IF .IFAB[IFB$V_RUP]
;  967          1028  2     THEN
;  968          1029  3         BEGIN
;  969          1030  3
;  970          1031  3         LOCAL
;  971          1032  3             SAVE_REC_ADDR;
;  972          1033  3
;  973          1034  3         SAVE_REC_ADDR = .REC_ADDR;
;  974          1035  3         REC_ADDR = .IRAB[IRB$L_OLDBUF] + 2;
;  975          1036  3
;  976        P 1037  3         RETURN_ON_ERROR (RM$RU_JOURNAL3 (RJR$_UPDATE,
;  977        P 1038  3                                          .IRAB[IRB$L_PUTUP_VBN],
;  978        P 1039  3                                          .IRAB[IRB$W_PUTUP_ID],
;  979        P 1040  3                                          .(.IRAB[IRB$L_OLDBUF])<0,16>),
;  980          1041  3                          RM$CLEAN_BDB());
;  981          1042  3
;  982          1043  3         REC_ADDR = .SAVE_REC_ADDR;
;  983          1044  3         END;
;  984          1045  2
;  985          1046  2     ! If any new alternate keys will have to be inserted as a result of this
;  986          1047  2     ! $UPDATE, then insert them at this time. To do so, first release the
;  987          1048  2     ! primary data bucket containing the old record, then insert all the
;  988          1049  2     ! alternate keys which need to be inserted, and finally re-position back to
;  989          1050  2     ! the old primary data record.
;  990          1051  2     !
;  991          1052  2     IF .BBLOCK[.IRAB[IRB$L_UPDBUF], UPD$V_INS_NEW]
;  992          1053  2     THEN
;  993          1054  2
;  994          1055  2 GETBACK:
;  995          1056  3         BEGIN
;  996          1057  3
;  997          1058  3         ! If RMS has performed lockabove optimization to lock the bucket from
;  998          1059  3         ! the level above, then release this bucket whose BDB has been saved in
;  999          1060  3         ! IRB$L_LOCK_BDB. This is because SIDR inserts will also try to perform
; 1000          1061  3         ! lockabove optimization.
; 1001          1062  3         !
; 1002          1063  3         IF TESTBITSC (IRAB[IRB$V_ABOVELCKD])
; 1003          1064  3         THEN
; 1004          1065  3             RELEASE (IRAB[IRB$L_LOCK_BDB]);
; 1005          1066  3
; 1006          1067  3         ! Release the primary data bucket containing the old record. Before the
; 1007          1068  3         ! bucket is released, the permanence bit is set so that RMS will
; 1008          1069  3         ! attempt to keep this bucket in the cache (it will eventually try to
```

```
;  1009          1070  3          ! re-position back to the old data record). The setting of the
;  1010          1071  3          ! permanence bit here should not ruin the setting of permanence for
;  1011          1072  3          ! alternate index root buckets with extra buffers because the bit
;  1012          1073  3          ! will be cleared when this bucket is reaccessed.
;  1013          1074  3          !
;  1014          1075  3          BDB = .IRAB[IRB$L_CURBDB];
;  1015          1076  3          IRAB[IRB$L_CURBDB] = 0;
;  1016          1077  3
;  1017          1078  3          IRAB[IRB$L_LAST_VBN] = .BDB[BDB$L_VBN];
;  1018          1079  3          BDB[BDB$V_PRM] = 1;
;  1019          1080  3          RM$RLSBKT(0);
;  1020          1081  3
;  1021          1082  3          ! If this is not a RU ROLLBACK Recovery operation then insert all the
;  1022          1083  3          ! alternate keys required by the new record.
;  1023          1084  3          !
;  1024          1085  3          IF NOT .IFAB[IFB$V_RU_RECVR]
;  1025          1086  3          THEN
;  1026          1087  4              RETURN_ON_ERROR (RM$INS_ALL_SIDR())
;  1027          1088  4
;  1028          1089  4          ! If this is a RU ROLLBACK Recovery operation then it is not necessary
;  1029          1090  4          ! to insert any new alternate keys since they already exist in the file.
;  1030          1091  4          ! It is only necessary to position to and un-delete them.
;  1031          1092  4          !
;  1032          1093  3          ELSE
;  1033          1094  4              BEGIN
;  1034          1095  4
;  1035          1096  4              ! Signal that the SIDRs positioned to are to be un-deleted, and
;  1036          1097  4              ! position REC_ADDR to the new record so that the alternate keys
;  1037          1098  4              ! maybe extracted to be used in the positioning.
;  1038          1099  4              !
;  1039          1100  4              IRAB[IRB$V_RU_UNDEL] = 1;
;  1040          1101  4              REC_ADDR = .IRAB[IRB$L_RBF];
;  1041          1102  4
;  1042          1103  4              ! Continue until all the alternate keys of the new record have been
;  1043          1104  4              ! un-deleted.
;  1044          1105  4              !
;  1045          1106  4              WHILE RM$GET_NEXT_KEY()
;  1046          1107  4              DO
;  1047          1108  4
;  1048          1109  4                  ! If there is a new SIDR to be un-deleted for this key of
;  1049          1110  4                  ! reference then do so.
;  1050          1111  4                  !
;  1051          1112  4                  IF .BBLOCK[.IRAB[IRB$L_UPDBUF] + .IDX_DFN[IDX$B_DESC_NO],
;  1052          1113  4                            UPD$V_INS_NEW]
;  1053          1114  4                  THEN
;  1054          1115  5                      BEGIN
;  1055          1116  5
;  1056          1117  5                      ! Extract the alternate key from the new record, and
;  1057          1118  5                      ! un-delete the corresponding SIDR.
;  1058          1119  5                      !
;  1059          1120  5                      AP = 3;
;  1060          1121  5                      RM$RECORD_KEY (KEYBUF_ADDR(2));
;  1061          1122  5                      RM$DELETE_SIDR();
;  1062          1123  4                      END;
;  1063          1124  4
;  1064          1125  4              IRAB[IRB$V_RU_UNDEL] = 0;
;  1065          1126  3              END;
```

```
;  1066          1127  3
;  1067          1128  3          ! Retrieve the primary key descriptor.
;  1068          1129  3          !
;  1069          1130  3          RM$KEY_DESC(0);
;  1070          1131  3
;  1071          1132  3          ! Clear IRB$V_ABOVELCKD as SIDR inserts may have left this bit set.
;  1072          1133  3          !
;  1073          1134  3          IRAB[IRB$V_ABOVELCKD] = 0;
;  1074          1135  3
;  1075          1136  3          ! Position back to the current primary data record so it maybe changed.
;  1076          1137  3          !
;  1077          1138  4          BEGIN
;  1078          1139  4
;  1079          1140  4          LOCAL
;  1080          1141  4              STATUS;
;  1081          1142  4
;  1082          1143  4          ! First, try to position back to the record using its last known
;  1083          1144  4          ! location. This should succeed unless the bucket containing the
;  1084          1145  4          ! current record has split, and the record has moved. The bit
;  1085          1146  4          ! IRB$V_NORLS_RNF is set, so that RMS can always clear the BDB's
;  1086          1147  4          ! permancence bit (set above before the bucket was released) regardless
;  1087          1148  4          ! of whether it was successful at position to the current record or not.
;  1088          1149  4          !
;  1089          1150  4          IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  1090          1151  4          IRAB[IRB$V_NORLS_RNF] = 1;
;  1091          1152  4
;  1092          1153  4          AP = .IRAB[IRB$W_LAST_ID];
;  1093          1154  4          STATUS = RM$FIND_BY_RFA (.IRAB[IRB$L_LAST_VBN]);
;  1094          1155  4
;  1095          1156  4          ! If RMS was able to find the current record in its last know location
;  1096          1157  4          ! verify, by comparing RFAs, that the record positioned to is in fact
;  1097          1158  4          ! the current record. If so, clear the BDB's permancance bit and
;  1098          1159  4          ! proceed with the update; otherwise, a second attempt will made to
;  1099          1160  4          ! position to the current record utilizing its RFA.
;  1100          1161  4          !
;  1101          1162  4          IF .STATUS
;  1102          1163  4          THEN
;  1103          1164  5              BEGIN
;  1104          1165  5
;  1105          1166  5              AP = 3;
;  1106          1167  5              IF .IRAB[IRB$W_PUTUP_ID] EQL RM$RECORD_ID()
;  1107          1168  5              THEN
;  1108          1169  5                  IF .IRAB[IRB$L_PUTUP_VBN] EQL RM$RECORD_VBN()
;  1109          1170  5                  THEN
;  1110          1171  6                      BEGIN
;  1111          1172  6                      BBLOCK[.IRAB[IRB$L_CURBDB], BDB$V_PRM] = 0;
;  1112          1173  6                      LEAVE GETBACK;
;  1113          1174  5                      END;
;  1114          1175  5              END
;  1115          1176  5
;  1116          1177  5          ! If RMS was unable to position to the current primary data record in
;  1117          1178  5          ! its last known location because of some obscure I/O error, return
;  1118          1179  5          ! that error; otherwise, a second attempt will be made to position to
;  1119          1180  5          ! the current record utilizing its RFA.
;  1120          1181  5          !
;  1121          1182  4          ELSE
;  1122          1183  6              IF NOT (.STATUS<0,16> EQL RMSERR(RNF)
```

```
;  1123          1184  5                                  or
;  1124          1185  5                      .STATUS<0,16> EQL RMSERR(DEL))
;  1125          1186  4              THEN
;  1126          1187  5                  BEGIN
;  1127          1188  5                  RM$CLEAN_BDB();
;  1128          1189  5                  RM$PUTUPD_ERROR();
;  1129          1190  5                  RETURN .STATUS;
;  1130          1191  4                  END;
;  1131          1192  3          END;
;  1132          1193  3
;  1133          1194  3          ! RMS was unable to locate the current primary data record in its
;  1134          1195  3          ! last known location although it was able to access the primary data
;  1135          1196  3          ! bucket in which it was last found. After clearing the permanence bit
;  1136          1197  3          ! in the BDB for this primary data bucket and releasing the bucket, RMS
;  1137          1198  3          ! attempts to position to the current primary data record by utilizing
;  1138          1199  3          ! the RFA of the current record.
;  1139          1200  3          !
;  1140          1201  3          BDB = .IRAB[IRB$L_CURBDB];
;  1141          1202  3          BDB[BDB$V_PRM] = 0;
;  1142          1203  3          RM$RLSBKT(0);
;  1143          1204  3          IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  1144          1205  3
;  1145        P 1206  3          RETURN_ON_ERROR
;  1146        P 1207  3              (RM$FIND_BY_RRV (.IRAB[IRB$L_PUTUP_VBN], .IRAB[IRB$W_PUTUP_ID], 0),
;  1147        P 1208  3                  BEGIN
;  1148        P 1209  3                  RM$CLEAN_BDB();
;  1149        P 1210  3                  RM$PUTUPD_ERROR();
;  1150          1211  3                  END);
;  1151          1212  3
;  1152          1213  2          END;              ! of block GETBACK
;  1153          1214  2
;  1154          1215  2      ! Having re-positioned to the current primary data record, RMS now replaces
;  1155          1216  2      ! the old version of the record with the new version performing any primary
;  1156          1217  2      ! data bucket splits and primary index updates that are required.
;  1157          1218  2      !
;  1158          1219  3      BEGIN
;  1159          1220  3
;  1160          1221  3      LOCAL
;  1161          1222  3          STATUS,
;  1162          1223  3          RECORD_SIZE,
;  1163          1224  3          RECOVH_SIZE;
;  1164          1225  3
;  1165          1226  3      ! Retrieve the address of the primary data bucket containing the current
;  1166          1227  3      ! primary data reccrd.
;  1167          1228  3      !
;  1168          1229  3      BDB = .IRAB[IRB$L_CURBDB];
;  1169          1230  3      BKT_ADDR = .BDB[BDB$L_ADDR];
;  1170          1231  3
;  1171          1232  3      ! Determine the number of bytes the new version of the record will take up
;  1172          1233  3      ! in the bucket, and the number of bytes the new version of the record will
;  1173          1234  3      ! take up in the bucket including record overhead. Both of these quantites
;  1174          1235  3      ! are prologue dependent, and the latter also depends upon the attributes
;  1175          1236  3      ! of the records in the file.
;  1176          1237  3      !   Also, reevaluate the record id for the record to be updated.  It may
;  1177          1238  3      ! easily have been displaced (as a result of a split, for example) since
;  1178          1239  3      ! IRB$W_LAST_ID was last saved.  We need this id in case the new record
;  1179          1240  3      ! will be larger than the old one, causing a delete + insert operation
```

```
;  1180          1241  3      ! to be performed.  If that happens, we need to use the same record id
;  1181          1242  3      ! for the newly inserted record to keep the same RFA.
;  1182          1243  3      !
;  1183          1244  3      !
;  1184          1245  3      IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  1185          1246  3      THEN
;  1186          1247  4          BEGIN
;  1187          1248  4          RECORD_SIZE = .IRAB[IRB$W_RSZ];
;  1188          1249  4
;  1189          1250  4          RECOVH_SIZE = .RECORD_SIZE + IRC$C_FIXOVHDSZ;
;  1190          1251  4          IF .IFAB[IFB$B_RFMORG] NEQU FAB$C_FIX
;  1191          1252  4          THEN
;  1192          1253  4              RECOVH_SIZE = .RECOVH_SIZE + IRC$C_DATSZFLD;
;  1193          1254  4          IRAB[IRB$W_LAST_ID] = .REC_ADDR[IRC$B_ID];
;  1194          1255  4          END
;  1195          1256  3      ELSE
;  1196          1257  4          BEGIN
;  1197          1258  4
;  1198          1259  4          ! The records in prologue 3 primary data buckets are packed. Record
;  1199          1260  4          ! packing includes key compression if it is enabled, and to insure that
;  1200          1261  4          ! the compression is done correctly, IRB$L_LST_NCMP is set to the last
;  1201          1262  4          ! known record with a zero front compressed primary key in the current
;  1202          1263  4          ! bucket.
;  1203          1264  4          !
;  1204          1265  4          IRAB[IRB$L_LST_NCMP] = .BKT_ADDR + BKT$C_OVERHDSZ;
;  1205          1266  4          RECORD_SIZE = RM$PACK_REC();
;  1206          1267  4
;  1207          1268  4          RECOVH_SIZE = .RECORD_SIZE + IRC$C_FIXOVHSZ3;
;  1208          1269  4          IF .IFAB[IFB$B_RFMORG] NEQU FAB$C_FIX
;  1209          1270  4              OR
;  1210          1271  5             (.IFAB[IFB$B_RFMORG] EQLU FAB$C_FIX
;  1211          1272  5                   AND
;  1212          1273  5                  .IDX_DFN[IDX$B_DATBKTYP] NEQU IDX$C_NCMPNCMP)
;  1213          1274  4          THEN
;  1214          1275  4              RECOVH_SIZE = .RECOVH_SIZE + IRC$C_DATSZFLD;
;  1215          1276  4          IRAB[IRB$W_LAST_ID] = .REC_ADDR[IRC$W_ID];
;  1216          1277  3          END;
;  1217          1278  3
;  1218          1279  3      ! Attempt to replace the old version of the current primary data record with
;  1219          1280  3      ! the new version without splitting the primary data bucket. The routine
;  1220          1281  3      ! RM$REPLACE will replace the old version with the new version if the latter
;  1221          1282  3      ! is the same size or smaller than the former. Failing that, it will delete
;  1222          1283  3      ! the old version of the primary data record, and RM$INSERT_UDR will try and
;  1223          1284  3      ! insert the new version into the primary data bucket.
;  1224          1285  3      !
;  1225          1286  4      IF NOT (STATUS = RM$REPLACE (.RECORD_SIZE))
;  1226          1287  3      THEN
;  1227          1288  4          BEGIN
;  1228          1289  4
;  1229          1290  4          ! If the size of the new record is smaller than the old record by one
;  1230          1291  4          ! byte and a specially formatted record must be built because the
;  1231          1292  4          ! process is currently within a Recovery Unit and the file is RU
;  1232          1293  4          ! Journalled, then increase the size of the record by two bytes to
;  1233          1294  4          ! allow for the specially formatted record to contain two record size
;  1234          1295  4          ! fields.
;  1235          1296  4          !
;  1236          1297  4          IF .IRAB[IRB$V_RU_UPDATE]
```

```
;  1237          1298  4          THEN
;  1238          1299  4              RECOVH_SIZE = .RECOVH_SIZE + IRC$C_DATSZFLD;
;  1239          1300  4
;  1240          1301  4          STATUS = RM$INSERT_UDR (RECOVH_SIZE);
;  1241          1302  3          END;
;  1242          1303  3
;  1243          1304  3      ! If RMS is successful at replacing the old version of the current primary
;  1244          1305  3      ! data record with the new version without causing a bucket split, then the
;  1245          1306  3      ! modified primary data bucket maybe marked dirty and released.
;  1246          1307  3      !
;  1247          1308  3      IF .STATUS
;  1248          1309  3      THEN
;  1249          1310  4          BEGIN
;  1250          1311  4          BDB[BDB$V_DRT] = 1;
;  1251          1312  4          IRAB[IRB$L_CURBDB] = 0;
;  1252          1313  4
;  1253        P 1314  4          RETURN_ON_ERROR (RM$RLSBKT (0),
;  1254        P 1315  4                                           BEGIN
;  1255        P 1316  4                                           RM$CLEAN_BDB();
;  1256        P 1317  4                                           RM$PUTUPD_ERROR();
;  1257          1318  4                                           END);
;  1258          1319  4
;  1259          1320  4          ! If an above level bucket was locked due to lockabove optimization,
;  1260          1321  4          ! then release it at this time.
;  1261          1322  4          !
;  1262          1323  4          IF (BDB = .IRAB[IRB$L_LOCK_BDB]) NEQU 0
;  1263          1324  4          THEN
;  1264          1325  5              BEGIN
;  1265          1326  5              RM$RLSBKT(0);
;  1266          1327  5              IRAB[IRB$L_LOCK_BDB] = 0;
;  1267          1328  4              END;
;  1268          1329  4          END
;  1269          1330  4
;  1270          1331  4          ! RMS was not able to replace the old version of the current primary
;  1271          1332  4          ! data record with the new version without splitting the primary data
;  1272          1333  4          ! bucket. Therefore, split the primary data bucket inserting the new
;  1273          1334  4          ! version of the current primary data record, and performing any index
;  1274          1335  4          ! updates that are required.
;  1275          1336  4          !
;  1276          1337  3      ELSE
;  1277          1338  4          BEGIN
;  1278          1339  4
;  1279          1340  4          ! Perform the split of the primary data level bucket and insert the new
;  1280          1341  4          ! version of the current primary data record.
;  1281          1342  4          !
;  1282          1343  4          IRAB[IRB$B_SPL_BITS] = 0;
;  1283          1344  4          IRAB[IRB$B_KEYSZ] = .IDX_DFN[IDX$B_KEYSZ];
;  1284          1345  4
;  1285        P 1346  4          RETURN_ON_ERROR (RM$PUT_UPD_SPL(.RECOVH_SIZE),
;  1286        P 1347  4                           BEGIN
;  1287        P 1348  4                           RM$CLEAN_BDB();
;  1288        P 1349  4                           RM$PUTUPD_ERROR();
;  1289          1350  4                           END);
;  1290          1351  4
;  1291          1352  4          ! Retrieve the BDB for the primary data bucket that has split, which is
;  1292          1353  4          ! still locked by this stream.
;  1293          1354  4          !
```

```
;  1294          1355  4          BDB = .IRAB[IRB$L_CURBDB];
;  1295          1356  4
;  1296          1357  4          ! If an index update is not required, just release the original primary
;  1297          1358  4          ! data bucket. The routine RM$PUT_UPD_SPL will clear the bit
;  1298          1359  4          ! IRB$V_UPDATE if an index update is not required following the primary
;  1299          1360  4          ! data bucket split, and set it if an index update is required.
;  1300          1361  4          !
;  1301          1362  4          IF TESTBITCS(IRAB[IRB$V_UPDATE])
;  1302          1363  4          THEN
;  1303          1364  5              BEGIN
;  1304          1365  5
;  1305          1366  5              IRAB[IRB$L_CURBDB] = 0;
;  1306          1367  5
;  1307        P 1368  5              RETURN_ON_ERROR (RM$RLSBKT(0), BEGIN
;  1308        P 1369  5                                              RM$CLEAN_BDB();
;  1309        P 1370  5                                              RM$PUTUPD_ERROR();
;  1310          1371  5                                              END);
;  1311          1372  5
;  1312          1373  5              ! If an above level bucket was locked due to lockabove optimization,
;  1313          1374  5              ! then release it at this time.
;  1314          1375  5              !
;  1315          1376  5              IF (BDB = .IRAB[IRB$L_LOCK_BDB]) NEQ 0
;  1316          1377  5              THEN
;  1317          1378  6                  BEGIN
;  1318          1379  6                  RM$RLSBKT(0);
;  1319          1380  6                  IRAB[IRB$L_LOCK_BDB] = 0;
;  1320          1381  5                  END;
;  1321          1382  5              END
;  1322          1383  5
;  1323          1384  5              ! The split of the primary data bucket requires updating of the primary
;  1324          1385  5              ! index. Perform the update at this time.
;  1325          1386  5              !
;  1326          1387  4              ELSE
;  1327          1388  5                  BEGIN
;  1328          1389  5
;  1329          1390  5                  IRAB[IRB$B_STOPLEVEL] = 1;
;  1330          1391  5                  IRAB[IRB$W_SRCHFLAGS] = IRB$M_POSINSERT;
;  1331          1392  5
;  1332          1393  6                  IF NOT (RAB[RAB$L_STV] = RM$INSS_OR_IDX())
;  1333          1394  5                  THEN
;  1334          1395  5                      IRAB[IRB$V_IDX_ERR] = 1;
;  1335          1396  5                  END
;  1336          1397  3              END;
;  1337          1398  2          END;
;  1338          1399  2
;  1339          1400  2          ! If there are alternate keys from the old record to be deleted, then the
;  1340          1401  2          ! old record has been saved in the IRB$L_OLDBUF record buffer. To delete
;  1341          1402  2          ! the old SIDRs, the alternate key of each SIDR to be deleted is in turn
;  1342          1403  2          ! extracted into keybuffer 2 and the SIDR deletion routine is called.
;  1343          1404  2          !
;  1344          1405  2          IF .BBLOCK[.IRAB[IRB$L_UPDBUF],UPD$V_OLD_DEL]
;  1345          1406  2          THEN
;  1346          1407  3              BEGIN
;  1347          1408  3
;  1348          1409  3              REC_ADDR = .IRAB[IRB$L_OLDBUF] + 2;
;  1349          1410  3
;  1350          1411  3              ! If the current operation is an UPDATE_IF, then the current record may
```

```
;  1351     1412  3          ! not be the record being updated. Yet, the routine RM$DELETE_SIDR
;  1352     1413  3          ! operates only on the current record; therefore, setup the record
;  1353     1414  3          ! being updated as the current record. Then is no need to restore these
;  1354     1415  3          ! fields because they will be zeroed out at the conclusion of the
;  1355     1416  3          ! operation anyway.
;  1356     1417  3          !
;  1357     1418  3          IF .IRAB[IRB$V_UPDATE_IF]
;  1358     1419  3          THEN
;  1359     1420  4              BEGIN
;  1360     1421  4              IRAB[IRB$L_UDR_VBN] = .IRAB[IRB$L_PUTUP_VBN];
;  1361     1422  4              IRAB[IRB$W_UDR_ID]  = .IRAB[IRB$W_PUTUP_ID];
;  1362     1423  3              END;
;  1363     1424  3
;  1364     1425  3          ! If this file is being Recovery Unit Journalled and the process is
;  1365     1426  3          ! currently within a recovery unit the set the IRB$V_RU_DELETE state
;  1366     1427  3          ! bit so that the SIDRs being deleted are only marked RU_DELETE and no
;  1367     1428  3          ! space is reclaimed.
;  1368     1429  3          !
;  1369     1430  3          IF .IFAB[IFB$V_RUP]
;  1370     1431  3          THEN
;  1371     1432  3              IRAB[IRB$V_RU_DELETE] = 1;
;  1372     1433  3
;  1373     1434  3          ! Loop through the index descriptors deleting those SIDRs that need to
;  1374     1435  3          ! be deleted.
;  1375     1436  3          !
;  1376     1437  3          WHILE RM$GET_NEXT_KEY()
;  1377     1438  3          DO
;  1378     1439  3
;  1379     1440  3              ! If this index has a SIDR which needs to be deleted, then delete
;  1380     1441  3              ! it.
;  1381     1442  3              !
;  1382     1443  3              IF .BBLOCK[.IRAB[IRB$L_UPDBUF] + .IDX_DFN[IDX$B_DESC_NO],
;  1383     1444  3                      UPD$V_OLD_DEL]
;  1384     1445  3              THEN
;  1385     1446  4                  BEGIN
;  1386     1447  4
;  1387     1448  4                  ! Extract the alternate key from the unpacked version of the
;  1388     1449  4                  ! old data record, and delete the corresponding SIDR.
;  1389     1450  4                  !
;  1390     1451  4                  AP = 3;
;  1391     1452  4                  RM$RECORD_KEY (KEYBUF_ADDR(2));
;  1392     1453  4                  RM$DELETE_SIDR();
;  1393     1454  3                  END;
;  1394     1455  3
;  1395     1456  3              IRAB[IRB$V_RU_DELETE] = 0;
;  1396     1457  2              END;
;  1397     1458  2
;  1398     1459  2          ! Having successfully completed the $UPDATE, finish up and return success.
;  1399     1460  2          !
;  1400     1461  2          RETURN RM$PUT_UPD_FIN()
;  1401     1462  2
;  1402     1463  1          END;
```

```
                                 34  BB 00000 RM$UPDATE3B::
                                                  PUSHR   #^M<R2,R4,R5>                   ; 0843
                              5E    04  C2 00002          SUBL2   #4, SP
                        0A    04 A9   13  E3 00005          BBCS    #19, 4(IRAB), 1$        ; 0995
                        06    04 A9   20  8A 0000A          BICB2   #32, 6(IRAB)            ; 1003
                                    0000G 30 0000E          BSBW    RM$UPDDELCOM            ; 1005
                              4E    50  E9 00011          BLBC    STATUS, 6$
                              06 00A1 CA  E8 00014 1$:      BLBS    161(IFAB), 2$           ; 1012
                                    0000G 30 00019          BSBW    RM$PUT_UPD_CHKS         ; 1014
                              40    50  E9 0001C          BLBC    STATUS, 5$
                                  FECA  30 0001F 2$:      BSBW    RM$UPDATE_SCAN          ; 1019
                              3A    50  E9 00022          BLBC    STATUS, 5$
                              03 00B7 CA  91 00025          CMPB    183(IFAB), #3           ; 1021
                                    06  1E 0002A          BGEQU   3$
                              50    01 A6  9A 0002C          MOVZBL  1(REC_ADDR), R0
                                    04  11 00030          BRB     4$
                              50    01 A6  3C 00032 3$:     MOVZWL  1(REC_ADDR), R0
                           74 A9    50  B0 00036 4$:     MOVW    R0, 116(IRAB)
                        27 00A2 CA      02  E1 0003A          BBC     #2, 162(IFAB), 8$       ; 1027
                              52        56  D0 00040          MOVL    REC_ADDR, SAVE_REC_ADDR ; 1034
                        56    6C A9    02  C1 00043          ADDL3   #2, 108(IRAB), REC_ADDR ; 1035
                              7E    6C  B9 3C 00048          MOVZWL  @108(IRAB), -(SP)       ; 1041
                              7E  0080  C9 3C 0004C          MOVZWL  128(IRAB), -(SP)
                                    78  A9 DD 00051          PUSHL   120(IRAB)
                                    1C  DD 00054          PUSHL   #28
                                    0000G 30 00056          BSBW    RM$RU_JOURNAL3
                              5E        10  C0 00059          ADDL2   #16, SP
                              05        50  E8 0005C          BLBS    STATUS, 7$
                                    0000G 30 0005F 5$:      BSBW    RM$CLEAN_BDB
                                    43  11 00062 6$:      BRB     11$
                              56        52  D0 00064 7$:     MOVL    SAVE_REC_ADDR, REC_ADDR ; 1043
                              03    64  B9 E8 00067 8$:     BLBS    @100(IRAB), 9$          ; 1052
                                  00F7  31 0006B          BRW     18$
                        11    04 A9   15  E5 0006E 9$:     BBCC    #21, 4(IRAB), 10$       ; 1063
                              54  0084  C9 D0 00073          MOVL    132(IRAB), BDB          ; 1065
                                  0084  C9 D4 00078          CLRL    132(IRAB)
                                    7E  D4 0007C          CLRL    -(SP)
                                    0000G 30 0007E          BSBW    RM$RLSBKT
                              5E        04  C0 00081          ADDL2   #4, SP
                              54    20  A9 D0 00084 10$:     MOVL    32(IRAB), BDB           ; 1075
                                    20  A9 D4 00088          CLRL    32(IRAB)                ; 1076
                           70 A9    1C  A4 D0 0008B          MOVL    28(BDB), 112(IRAB)      ; 1078
                           0A A4        08 88 00090          BISB2   #8, 10(BDB)             ; 1079
                                        7E D4 00094          CLRL    -(SP)                   ; 1080
                                    0000G 30 00096          BSBW    RM$RLSBKT
                              5E        04  C0 00099          ADDL2   #4, SP
                              09 00A1 CA  E8 0009C          BLBS    161(IFAB), 12$          ; 1085
                                    0000G 30 000A1          BSBW    RM$INS_ALL_SIDR         ; 1087
                              39        50  E8 000A4          BLBS    STATUS, 15$
                                  01DC  31 000A7 11$:     BRW     37$
                           07 A9    40  8F 88 000AA 12$:     BISB2   #64, 7(IRAB)            ; 1100
                              56    58  A9 D0 000AF          MOVL    88(IRAB), REC_ADDR      ; 1101
                                    0000G 30 000B3 13$:     BSBW    RM$GET_NEXT_KEY         ; 1106
                              22        50  E9 000B6          BLBC    R0, 14$
                              50    10  A7 9A 000B9          MOVZBL  16(IDX_DFN), R0         ; 1113
                              50    64  A9 C0 000BD          ADDL2   100(IRAB), R0
                              EF        60  E9 000C1          BLBC    (R0), 13$
```

```
                                5C      03   D0 000C4          MOVL    #3, AP                                  ; 1120
                                50  00B4   CA   3C 000C7      MOVZWL  180(IFAB), R0                           ; 1121
                                    60 B940   9F   000CC      PUSHAB  @96(IRAB)[R0]                           ;
                                  0000G 30    000D0          BSBW    RM$RECORD_KEY                           ;
                                5E      04   C0 000D3          ADDL2   #4, SP                              ;
                                  0000G 30    000D6          BSBW    RM$DELETE_SIDR                          ; 1122
                                    D8  11    000D9          BRB     13$                                     ; 1112
                          07    A9      40   8F   8A 000DB 14$:   BICB2   #64, 7(IRAB)                      ; 1125
                                      7E   D4 000E0 15$:       CLRL    -(SP)                                   ; 1130
                                  0000G 30    000E2          BSBW    RM$KEY_DESC                             ;
                          06    A9      20   8A 000E5          BICB2   #32, 6(IRAB)                        ; 1134
                          40    A9      01   90 000E9          MOVB    #1, 64(IRAB)                        ; 1150
                          42    A9      20   88 000ED          BISB2   #32, 66(IRAB)                       ; 1151
                                5C  74  A9   3C 000F1          MOVZWL  116(IRAB), AP                       ; 1153
                                6E  70  A9   D0 000F5          MOVL    112(IRAB), (SP)                     ; 1154
                                  0000G 30    000F9          BSBW    RM$FIND_BY_RFA                          ;
                                5E      04   C0 000FC          ADDL2   #4, SP                              ;
                                52      50   D0 000FF          MOVL    R0, STATUS                          ;
                                22      52   E9 00102          BLBC    STATUS, 16$                         ; 1162
                                5C      03   D0 00105          MOVL    #3, AP                              ; 1166
                                  0000G 30    00108          BSBW    RM$RECORD_ID                            ; 1167
     50     0080    C9                10      00   ED 0010B          CMPZV   #0, #16, T28(IRAB), R0          ;
                                          2D   12 00112          BNEQ    17$                                 ;
                                  0000G 30    00114          BSBW    RM$RECORD_VBN                           ; 1169
                                50  78  A9   D1 00117          CMPL    120(IRAB), R0                       ;
                                          24   12 0011B          BNEQ    17$                                 ;
                                50  20  A9   D0 0011D          MOVL    32(IRAB), R0                        ; 1172
                          0A    A0      08   8A 00121          BICB2   #8, 10(R0)                          ;
                                          3E   11 00125          BRB     18$                                 ; 1173
                        82B2    8F      52   B1 00127 16$:      CMPW    STATUS, #33458                      ; 1183
                                          13   13 0012C          BEQL    17$                                 ;
                        8262    8F      52   B1 0012E          CMPW    STATUS, #33378                      ; 1185
                                          0C   13 00133          BEQL    17$                                 ;
                                  0000G 30    00135          BSBW    RM$CLEAN_BDB                            ; 1188
                                  0000G 30    00138          BSBW    RM$PUTUPD_ERROR                         ; 1189
                                50      52   D0 0013B          MOVL    STATUS, R0                          ; 1190
                                  0145  31    0013E          BRW     37$                                     ;
                                54      20  A9   D0 00141 17$:      MOVL    32(IRAB), BDB                   ; 1201
                          0A    A4      08   8A 00145          BICB2   #8, 10(BDB)                         ; 1202
                                      7E   D4 00149          CLRL    -(SP)                                   ; 1203
                                  0000G 30    0014B          BSBW    RM$RLSBKT                               ;
                          40    A9      01   90 0014E          MOVB    #1, 64(IRAB)                        ; 1204
                                      6E   D4 00152          CLRL    (SP)                                    ; 1211
                                7E  0080    C9   3C 00154          MOVZWL  128(IRAB), -(SP)                    ;
                                  78  A9   DD 00159          PUSHL   120(IRAB)                               ;
                                  0000G 30    0015C          BSBW    RM$FIND_BY_RRV                          ;
                                5E      0C   C0 0015F          ADDL2   #12, SP                             ;
                                7D      50   E9 00162          BLBC    STATUS, 27$                         ;
                                54  20      A9   D0 00165 18$:      MOVL    32(IRAB), BDB                   ; 1229
                                55  18      A4   D0 00169          MOVL    24(BDB), BKT_ADDR               ; 1230
                                03  00B7    CA   91 0016D          CMPB    183(IFAB), #3                   ; 1245
                                          18   1E 00172          BGEQU   20$                                 ;
                                50  56      A9   3C 00174          MOVZWL  86(IRAB), RECORD_SIZE           ; 1248
                                6E  07      A0   9E 00178          MOVAB   7(R0), RECOVH_SIZE              ; 1250
                                01  50      AA   91 0017C          CMPB    80(IFAB), #1                    ; 1251
                                          03   13 00180          BEQL    19$                                 ;
                                6E      02   C0 00182          ADDL2   #2, RECOVH_SIZE                     ; 1253
```

```
                                74   A9      01   A6  9B 00185 19$:    MOVZBW  1(REC_ADDR), 116(IRAB)          ;1254
                                                 21  11 0018A         BRB     23$                            ;1245
                              0098   C9      0E   A5  9E 0018C 20$:    MOVAB   14(R5), 152(IRAB)              ;1265
                                                0000G 30 00192         BSBW    RM$PACK_REC                    ;1266
                                     6E      09   A0  9E 00195         MOVAB   9(R0), RECOVH_SIZE             ;1268
                                     01      50   AA  91 00199         CMPB    80(IFAB), #1                   ;1269
                                                 06  12 0019D         BNEQ    21$
                                     06      29   A7  91 0019F         CMPB    41(IDX_DFN), #6                ;1273
                                                 03  13 001A3         BEQL    22$
                                     6E          02  C0 001A5 21$:     ADDL2   #2, RECOVH_SIZE                ;1275
                                   74    A9      01   A6  B0 001A8 22$:    MOVW    1(REC_ADDR), 116(IRAB)         ;1276
                                                 50  DD 001AD 23$:     PUSHL   RECORD_SIZE                    ;1286
                                               FC17  30 001AF         BSBW    RM$REPLACE
                                     5E          04  C0 001B2         ADDL2   #4, SP
                                     13          50  E8 001B5         BLBS    STATUS, 25$
                                               07    A9  95 001B8         TSTB    7(IRAB)                        ;1297
                                                 03  18 001BB         BGEQ    24$
                                     6E          02  C0 001BD         ADDL2   #2, RECOVH_SIZE                ;1299
                                                 5E  DD 001C0 24$:     PUSHL   SP                             ;1301
                                               0000G 30 001C2         BSBW    RM$INSERT_UDR
                                     5E          04  C0 001C5         ADDL2   #4, SP
                                     06          50  E9 001C8         BLBC    STATUS, 26$                    ;1308
                                   0A    A4          02  88 001CB 25$:     BISB2   #2, 10(BDB)                    ;1311
                                                 1D  11 001CF         BRB     28$                            ;1312
                                               44    A9  94 001D1 26$:     CLRB    68(IRAB)                       ;1343
                                 00A6   C9      20   A7  90 001D4         MOVB    32(IDX_DFN), 166(IRAB)         ;1344
                                                 6E  DD 001DA         PUSHL   RECOVH_SIZE                    ;1350
                                               0000G 30 001DC         BSBW    RM$PUT_UPD_SPL
                                     5E          04  C0 001DF         ADDL2   #4, SP
                                     17          50  E9 001E2 27$:     BLBC    STATUS, 29$
                                     54    20    A9  D0 001E5         MOVL    32(IRAB), BDB                  ;1355
                                 2C      04    A9      13  E2 001E9         BBSS    #19, 4(IRAB), 31$              ;1362
                                                 20  A9  D4 001EE 28$:     CLRL    32(IRAB)                       ;1366
                                                 7E  D4 001F1         CLRL    -(SP)                          ;1371
                                               0000G 30 001F3         BSBW    RM$RLSBKT
                                     5E          04  C0 001F6         ADDL2   #4, SP
                                     09          50  E8 001F9         BLBS    STATUS, 30$
                                               0000G 30 001FC 29$:     BSBW    RM$CLEAN_BDB
                                               0000G 30 001FF         BSBW    RM$PUTUPD_ERROR
                                               0081  31 00202         BRW     37$
                                     54      0084   C9  D0 00205 30$:    MOVL    132(IRAB), BDB                 ;1376
                                                 24  13 0020A         BEQL    32$
                                                 7E  D4 0020C         CLRL    -(SP)                          ;1379
                                               0000G 30 0020E         BSBW    RM$RLSBKT
                                     5E          04  C0 00211         ADDL2   #4, SP
                                               0084   C9  D4 00214         CLRL    132(IRAB)                      ;1380
                                                 16  11 00218         BRB     32$                            ;1362
                                   41    A9          01  90 0021A 31$:     MOVB    #1, 65(IRAB)                   ;1390
                                   42    A9          01  B0 0021E         MOVW    #1, 66(IRAB)                   ;1391
                                               0000G 30 00222         BSBW    RM$INSS_OR_IDX                 ;1393
                                   0C    A8          50  D0 00225         MOVL    R0, 12(RAB)
                                     04          50  E8 00229         BLBS    R0, 32$
                                   06    A9          02  88 0022C         BISB2   #2, 6(IRAB)                    ;1395
                             4E    64    B9          01  E1 00230 32$:     BBC     #1, @100(IRAB), 36$            ;1405
                             56    6C    A9          02  C1 00235         ADDL3   #2, 108(IRAB), REC_ADDR        ;1409
                             0D    06    A9          04  E1 0023A         BBC     #4, 6(IRAB), 33$               ;1418
                                 00B0   C9      78   A9  D0 0023F         MOVL    120(IRAB), 176(IRAB)           ;1421
```

```
                00BC   C9     0080    C9  B0 00245          MOVW    128(IRAB), 188(IRAB)                    ; 1422
          04    00A2   CA             02  E1 0024C 33$:     BBC     #2, 162(IFAB), 34$                      ; 1430
           07    A9             20  88 00252              BISB2   #32, 7(IRAB)                            ; 1432
                        0000G  30 00256 34$:     BSBW    RM$GET_NEXT_KEY                         ; 1437
                23             50  E9 00259              BLBC    R0, 35$
                50       10    A7  9A 0025C              MOVZBL  16(IDX_DFN), R0                 ; 1444
                50       64    A9  C0 00260              ADDL2   100(IRAB), R0
          EE    60             01  E1 00264              BBC     #1, (R0), 34$
                5C             03  D0 00268              MOVL    #3, AP                          ; 1451
                50     00B4    CA  3C 0026B              MOVZWL  180(IFAB), R0                   ; 1452
                     60 B940   9F  00270              PUSHAB  @96(IRAB)[R0]
                        0000G  30 00274              BSBW    RM$RECORD_KEY
                5E             04  C0 00277              ADDL2   #4, SP
                        0000G  30 0027A              BSBW    RM$DELETE_SIDR                  ; 1453
                               D7  11 0027D              BRB     34$                             ; 1443
           07    A9             20  8A 0027F 35$:     BICB2   #32, 7(IRAB)                    ; 1456
                        0000G  30 00283 36$:     BSBW    RM$PUT_UPD_FIN                  ; 1461
                5E             04  C0 00286 37$:     ADDL2   #4, SP                          ; 1463
                               34  BA 00289              POPR    #^M<R2,R4,R5>
                                   05 0028B              RSB

; Routine Size:  652 bytes,    Routine Base:  RM$RMS3 + 0237


; 1403          1464  1
; 1404          1465  1 END
; 1405          1466  1
; 1406          1467  0 ELUDOM



                                PSECT SUMMARY

     Name                       Bytes                          Attributes

RM$RMS3                          1219  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       107          3       154         00:00.4


                                COMMAND QUALIFIERS
```

```
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3UPDATE/OBJ=OBJ$:RM3UPDATE MSRC$:RM3UPDATE/UPDATE=(ENH$:RM3UPDATE)

; Size:         1219 code + 0 data bytes
; Run Time:       00:32.6
; Elapsed Time:   01:09.9
; Lines/CPU Min:   2702
; Lexemes/CPU-Min: 16349
; Memory Used:  286 pages
; Compilation Complete
```

RM3SSIDR
LIS

RM3UPSIDX
LIS

RM3SRCHKY
LIS

RM3UPDDEL
LIS

RM3UPDATE
LIS