```
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
```

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

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

RM3
V04

```
;     1          0001  0 MODULE RM3PUT (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 !
;    32          0032  1 ! FACILITY:     RMS32 Index Sequential File Organization
;    33          0033  1 !
;    34          0034  1 ! ABSTRACT:
;    35          0035  1 !                The high level $PUT code.
;    36          0036  1 !
;    37          0037  1 !
;    38          0038  1 ! ENVIRONMENT:
;    39          0039  1 !
;    40          0040  1 !                VAX/VMS Operating System
;    41          0041  1 !
;    42          0042  1 !--
;    43          0043  1
;    44          0044  1 !
;    45          0045  1 ! AUTHOR:         Todd M. Katz          RE-CREATION DATE:        09-Jan-83
;    46          0046  1 !
;    47          0047  1 ! MODIFIED BY:
;    48          0048  1 !
;    49          0049  1 !       V03-022 DGB0071         Donald G. Blair         17-Jul-1984
;    50          0050  1 !               Return STV correctly after disk-quota-exceeded error
;    51          0051  1 !               during root bucket split.
;    52          0052  1 !
;    53          0053  1 !       V03-021 DAS0002         David Solomon           25-Mar-1984
;    54          0054  1 !               Fix broken branches to RM$RU_JOURNAL3 and RM$RU_REFORMAT.
;    55          0055  1 !
;    56          0056  1 !       V03-020 DGB0032         Donald G. Blair         20-Mar-1984
;    57          0057  1 !               Fix broken branch to RM$UNLOCK
```

```
;    58          0058  1 !
;    59          0059  1 !       V03-019 DAS0001         David Solomon           05-Mar-1984
;    60          0060  1 !               Don't set IRB$V_UNLOCK_RP if ULK set (FT1 QAR 628).
;    61          0061  1 !
;    62          0062  1 !       V03-018 TSK0001         Tamar Krichevsky        22-Jun-1983
;    63          0063  1 !               Fix broken branch to RM$LOCK.
;    64          0064  1 !
;    65          0065  1 !       V03-017 TMK0012         Todd M. Katz            26-Mar-1983
;    66          0066  1 !               Change the linkage for RM$RU_JOURNAL3 from RL$RABREG_467 to
;    67          0067  1 !               RL$RABREG_67.
;    68          0068  1 !
;    69          0069  1 !       V03-016 MCN0002         Maria del C. Nasr       24-Mar-1983
;    70          0070  1 !               More linkages reorganization.
;    71          0071  1 !
;    72          0072  1 !       V03-015 TMK0011         Todd M. Katz            16-Mar-1983
;    73          0073  1 !               Change the linkage for RM$RU_JOURNAL3 from RL$RABREG_67 to
;    74          0074  1 !               RL$RABREG_467.
;    75          0075  1 !
;    76          0076  1 !       V03-014 TMK0010         Todd M. Katz            16-Mar-1983
;    77          0077  1 !               Change the symbol RMSR$_PUT tp RJR$_PUT.
;    78          0078  1 !
;    79          0079  1 !       V03-013 TMK0009         Todd M. Katz            11-Mar-1983
;    80          0080  1 !               Set the bit CSH$V_LOCK in the field IRB$B_CACHEFLGS before
;    81          0081  1 !               calling RM$GET_BKT within RM$CONFIRM_EOF so that the bucket
;    82          0082  1 !               retrieved will be exclusively locked.
;    83          0083  1 !
;    84          0084  1 !       V03-012 MCN0001         Maria del C. Nasr       28-Feb-1983
;    85          0085  1 !               Reorganize linkages
;    86          0086  1 !
;    87          0087  1 !       V03-011 TMK0008         Todd M. Katz            19-Dec-1982
;    88          0088  1 !               Re-write the routines within this module adding support for
;    89          0089  1 !               Recovery Unit Journalling of $PUT.
;    90          0090  1 !
;    91          0091  1 !*****
;    92          0092  1 !
;    93          0093  1
;    94          0094  1 LIBRARY 'RMSLIB:RMS';
;    95          0095  1
;    96          0096  1 REQUIRE 'RMSSRC:RMSIDXDEF';
;    97          0161  1
;    98          0162  1 ! Define default PSECTs for code.
;    99          0163  1 !
;   100          0164  1 PSECT
;   101          0165  1     CODE = RM$RMS3(PSECT_ATTR),
;   102          0166  1     PLIT = RM$RMS3(PSECT_ATTR);
;   103          0167  1
;   104          0168  1 ! Linkages.
;   105          0169  1 !
;   106          0170  1 LINKAGE
;   107          0171  1     L_COMPARE_KEY,
;   108          0172  1     L_ERROR_LINK1,
;   109          0173  1     L_PRESERVE1,
;   110          0174  1     L_QUERY_AND_LOCK,
;   111          0175  1     L_RABREG,
;   112          0176  1     L_RABREG_4567,
;   113          0177  1     L_RABREG_457,
;   114          0178  1     L_RABREG_567,
```

```
;  115          0179  1      L_RABREG_67,
;  116          0180  1      L_RABREG_7,
;  117          0181  1
;  118          0182  1      ! Local Routine Linkage
;  119          0183  1      !
;  120          0184  1      RL$LINKAGE = JSB () :
;  121          0185  1                   GLOBAL (COMMON_RABREG, R_IDX_DFN, R_REC_ADDR);
;  122          0186  1
;  123          0187  1  ! Forward Routines.
;  124          0188  1  !
;  125          0189  1  FORWARD ROUTINE
;  126          0190  1      RM$UPDATE_IF          : RL$LINKAGE;
;  127          0191  1
;  128          0192  1  ! External Routines.
;  129          0193  1  !
;  130          0194  1  EXTERNAL ROUTINE
;  131          0195  1      RM$BKT_SORT           : RL$RABREG_7,
;  132          0196  1      RM$CLEAN_BDB          : RL$ERROR_LINK1,
;  133          0197  1      RM$COMPARE_KEY        : RL$COMPARE_KEY,
;  134          0198  1      RM$CSEARCH_TREE       : RL$RABREG_67,
;  135          0199  1      RM$DELETE_ODR         : RL$RABREG_4567,
;  136          0200  1      RM$GETBKT             : RL$RABREG_457,
;  137          0201  1      RM$GETNEXT_REC        : RL$RABREG_67,
;  138          0202  1      RM$INS_ALL_SIDR       : RL$RABREG_4567,
;  139          0203  1      RM$INSERT_ODR         : RL$RABREG_4567,
;  140          0204  1      RM$INSS_OR_IDX        : RL$RABREG_567,
;  141          0205  1      RM$LOCK               : RL$QUERY_AND_LOCK ADDRESSING_MODE( LONG_RELATIVE ),
;  142          0206  1      RM$MAKE_INDEX         : RL$RABREG_7,
;  143          0207  1      RM$PACK_REC           : RL$RABREG_567,
;  144          0208  1      RM$PUT_OPD_CHKS       : RL$RABREG_7,
;  145          0209  1      RM$PUT_UPD_FIN        : RL$RABREG,
;  146          0210  1      RM$PUT_UPD_SPL        : RL$RABREG_4567,
;  147          0211  1      RM$RECORD_KEY         : RL$PRESERVE1,
;  148          0212  1      RM$RECORD_ID          : RL$RABREG_67,
;  149          0213  1      RM$RECORD_VBN         : RL$PRESERVE1,
;  150          0214  1      RM$RLSBKT             : RL$PRESERVE1,
;  151          0215  1      RM$RU_JOURNAL3        : RL$RABREG_67 ADDRESSING_MODE( LONG_RELATIVE ),
;  152          0216  1      RM$RU_REFORMAT        : RL$RABREG_567 ADDRESSING_MODE( LONG_RELATIVE ),
;  153          0217  1      RM$UNLOCK             : RL$QUERY_AND_LOCK ADDRESSING_MODE( LONG_RELATIVE ),
;  154          0218  1      RM$UPDATE3B           : RL$RABREG_67;
```

```
;  156     0219  1 %SBTTL 'RM$CONFIRM_EOF'
;  157     0220  1 ROUTINE RM$CONFIRM_EOF : RL$LINKAGE =
;  158     0221  1
;  159     0222  1 !++
;  160     0223  1 !
;  161     0224  1 ! FUNCTIONAL DESCRIPTION:
;  162     0225  1 !
;  163     0226  1 !       On position to EOF, while doing the first PUT, sequentially scan
;  164     0227  1 !       the file to verify there are no non-deleted records following the
;  165     0228  1 !       position of insert.
;  166     0229  1 !
;  167     0230  1 ! CALLING SEQUENCE:
;  168     0231  1 !       RM$CONFIRM_EOF()
;  169     0232  1 !
;  170     0233  1 ! INPUT PARAMETERS:
;  171     0234  1 !       NONE
;  172     0235  1 !
;  173     0236  1 ! IMPLICIT INPUTS:
;  174     0237  1 !
;  175     0238  1 !       IDX_DFN                 - address of the primary key descriptor
;  176     0239  1 !            IDX$B_DATBKTSZ     - size of the primary data bucket in blocks
;  177     0240  1 !
;  178     0241  1 !       IRAB                    - address of IRAB
;  179     0242  1 !           IRB$V_ABOVELCKD     - if set, lockabove optimization is in effect
;  180     0243  1 !           IRB$L_CURBDB        - address of BDB for insertion bucket
;  181     0244  1 !           IRB$B_CUR_KREF      - current NRP key of reference
;  182     0245  1 !           IRB$L_LOCK_BDB      - address of BDB of above locked bucket
;  183     0246  1 !           IRB$L_MIDX_TMP3     - VBN of bucket containing insertion position
;  184     0247  1 !
;  185     0248  1 !       REC_ADDR                -  address of position of insertion
;  186     0249  1 !
;  187     0250  1 ! OUTPUT PARAMETERS:
;  188     0251  1 !       NONE
;  189     0252  1 !
;  190     0253  1 ! IMPLICIT OUTPUTS:
;  191     0254  1 !
;  192     0255  1 !       IRAB                    - address of IRAB
;  193     0256  1 !           IRB$V_ABOVELCKD     - 0
;  194     0257  1 !           IRB$L_CURBDB        - address of BDB for starting data bucket
;  195     0258  1 !           IRB$L_LOCK_BDB      - 0
;  196     0259  1 !
;  197     0260  1 !       REC_ADDR                - address of record to start re-positioning with
;  198     0261  1 !
;  199     0262  1 ! ROUTINE VALUE:
;  200     0263  1 !
;  201     0264  1 !       ROP - if secondary key selected
;  202     0265  1 !       SUC - if record is really EOF record
;  203     0266  1 !       SEQ - is record isn't EOF record
;  204     0267  1 !
;  205     0268  1 ! SIDE EFFECTS:
;  206     0269  1 !
;  207     0270  1 !       On all errors, all locked buckets are released.
;  208     0271  1 !       Otherwise, RMS returns positioned to the first record in the original
;  209     0272  1 !           bucket.
;  210     0273  1 !
;  211     0274  1 !--
;  212     0275  1
```

```
;  213     0276  2      BEGIN
;  214     0277  2
;  215     0278  2      EXTERNAL REGISTER
;  216     0279  2          COMMON_RAB_STR,
;  217     0280  2          R_IDX_DFN_STR,
;  218     0281  2          R_REC_ADDR_STR;
;  219     0282  2
;  220     0283  2      GLOBAL REGISTER
;  221     0284  2          COMMON_IO_STR;
;  222     0285  2
;  223     0286  2      ! Position to EOF not supported for secondary keys. Return the appropriate
;  224     0287  2      ! error.
;  225     0288  2      !
;  226     0289  2      IF .IRAB[IRB$B_CUR_KREF] NEQU 0
;  227     0290  2      THEN
;  228     0291  2          RETURN RMSERR(ROP);
;  229     0292  2
;  230     0293  2      ! Release the level one index bucket if lock above optimization was
;  231     0294  2      ! utilized.
;  232     0295  2      !
;  233     0296  2      IF TESTBITSC (IRAB[IRB$V_ABOVELCKD])
;  234     0297  2      THEN
;  235     0298  2          RELEASE (IRAB[IRB$L_LOCK_BDB]);
;  236     0299  2
;  237     0300  2      ! Retrieve the the address of the primary data bucket.
;  238     0301  2      !
;  239     0302  2      BDB = .IRAB [IRB$L_CURBDB];
;  240     0303  2      BKT_ADDR = .BDB[BDB$L_ADDR];
;  241     0304  2
;  242     0305  2      ! Scan this and subsequent primary data level buckets in the horizontal
;  243     0306  2      ! chain starting with the record following the point of insertion of the
;  244     0307  2      ! new record in order to prove that there are no records following the one
;  245     0308  2      ! which will be inserted.
;  246     0309  2      !
;  247     0310  2      DO
;  248     0311  3          BEGIN
;  249     0312  3
;  250     0313  3          LOCAL
;  251     0314  3              END_OF_BUCKET,
;  252     0315  3              NEXT_BUCKET;
;  253     0316  3
;  254     0317  3          ! If the current bucket is not a primary data bucket then signal a
;  255     0318  3          ! bugcheck.
;  256     0319  3          !
;  257     0320  3          IF .BKT_ADDR[BKT$B_LEVEL] NEQU 0
;  258     0321  3          THEN
;  259     0322  3              BUG_CHECK;
;  260     0323  3
;  261     0324  3          ! Obtain the address of the end of the bucket.
;  262     0325  3          !
;  263     0326  3          END_OF_BUCKET = .BKT_ADDR + .BKT_ADDR[BKT$W_FREESPACE];
;  264     0327  3
;  265     0328  3          ! Determine whether there are any non-deleted records in this bucket.
;  266     0329  3          !
;  267     0330  3          WHILE .REC_ADDR LSSA .END_OF_BUCKET
;  268     0331  3                AND
;  269     0332  3               NOT .REC_ADDR[IRC$V_RRV]
```

```
;  270     0333  3          DO
;  271     0334  3
;  272     0335  3              ! If the current record is not deleted, then released all locked
;  273     0336  3              ! buckets and return an error.
;  274     0337  3              !
;  275     0338  3              IF  NOT .REC_ADDR[IRC$V_DELETED]
;  276     0339  3                   AND
;  277     0340  3                  NOT .REC_ADDR[IRC$V_RU_DELETE]
;  278     0341  3              THEN
;  279     0342  4                  BEGIN
;  280     0343  4                  RM$CLEAN_BDB();
;  281     0344  4                  RETURN RMSERR(SEQ);
;  282     0345  4                  END
;  283     0346  4
;  284     0347  4              ! Otherwise, position to the next record in the bucket.
;  285     0348  4              !
;  286     0349  3              ELSE
;  287     0350  3                  RM$GETNEXT_REC();
;  288     0351  3
;  289     0352  3          ! If this primary data bucket is the last bucket in the file, then
;  290     0353  3          ! the record RMS is inserting is not followed by any non-deleted records
;  291     0354  3          ! in the file, so return success.
;  292     0355  3          !
;  293     0356  3          IF .BKT_ADDR[BKT$V_LASTBKT]
;  294     0357  3          THEN
;  295     0358  4              BEGIN
;  296     0359  4
;  297     0360  4              ! If RMS had to scan multiple buckets, then it no longer has
;  298     0361  4              ! accessed the bucket containing the point of insertion of the new
;  299     0362  4              ! record. Release the current bucket, and reaccess this original
;  300     0363  4              ! bucket, so that RMS can start its re-positioning with the first
;  301     0364  4              ! record in this bucket.
;  302     0365  4              !
;  303     0366  4              IF .BDB[BDB$L_VBN] NEQU .IRAB[IRB$L_MIDX_TMP3]
;  304     0367  4              THEN
;  305     0368  5                  BEGIN
;  306     0369  5                  IRAB[IRB$L_CURBDB] = 0;
;  307     0370  5                  RM$RLSBKT(0);
;  308     0371  5
;  309     0372  5                  IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  310   P 0373  5                  RETURN_ON_ERROR (RM$GETBKT (.IRAB[IRB$L_MIDX_TMP3],
;  311     0374  5                                              .IDX_DFN[IDX$B_DATBKTSZ] * 512));
;  312     0375  5                  IRAB[IRB$L_CURBDB] = .BDB;
;  313     0376  4                  END;
;  314     0377  4
;  315     0378  4              ! RMS will re-position to the point of insertion of the new record
;  316     0379  4              ! starting with the first record in the bucket.
;  317     0380  4              !
;  318     0381  4              REC_ADDR = .BKT_ADDR + BKT$C_OVERHDSZ;
;  319     0382  4              RETURN RMSSUC(SOC);
;  320     0383  3              END;
;  321     0384  3
;  322     0385  3          ! This current bucket is not the last bucket in the file; therefore,
;  323     0386  3          ! save the VBN of the next primary data bucket in the horizontal chain,
;  324     0387  3          ! release the current bucket, access the next bucket, and continue the
;  325     0388  3          ! scan for non-deleted primary data records with the first record in
;  326     0389  3          ! that bucket.
```

```
;  327     0390  3          !
;  328     0391  3          NEXT_BUCKET = .BKT_ADDR[BKT$L_NXTBKT];
;  329     0392  3
;  330     0393  3          IRAB[IRB$L_CURBDB] = 0;
;  331     0394  3          RM$RLSBKT(0);
;  332     0395  3
;  333     0396  3          IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  334   P 0397  3          RETURN_ON_ERROR (RM$GETBKT (.NEXT_BUCKET,
;  335     0398  3                                      .IDX_DFN[IDX$B_DATBKTSZ] * 512));
;  336     0399  3
;  337     0400  3          IRAB[IRB$L_CURBDB] = .BDB;
;  338     0401  3          REC_ADDR = .BKT_ADDR + BKT$C_OVERHDSZ;
;  339     0402  3          END
;  340     0403  2      UNTIL 0;
;  341     0404  2
;  342     0405  2      RETURN RMSSUC(SUC);
;  343     0406  1      END;
```

```
                                .TITLE  RM3PUT
                                .IDENT  \V04-000\

                                .EXTRN  RM$BKT_SORT, RM$CLEAN_BDB
                                .EXTRN  RM$COMPARE_KEY, RM$CSEARCH_TREE
                                .EXTRN  RM$DELETE_ODR, RM$GETBKT
                                .EXTRN  RM$GETNEXT_REC, RM$INS_ALL_SIDR
                                .EXTRN  RM$INSERT_ODR, RM$INSS_OR_IDX
                                .EXTRN  RM$LOCK, RM$MAKE_INDEX
                                .EXTRN  RM$PACK_REC, RM$PUT_UPD_CHKS
                                .EXTRN  RM$PUT_OPD_FIN, RM$PUT_OPD_SPL
                                .EXTRN  RM$RECORD_KEY, RM$RECORD_ID
                                .EXTRN  RM$RECORD_VBN, RM$RLSBKT
                                .EXTRN  RM$RU_JOURNAL3, RM$RU_REFORMAT
                                .EXTRN  RM$UNLOCK, RM$UPDATE3B
                                .EXTRN  RM$BUG3

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

                          34    BB 00000 RM$CONFIRM_EOF:
                                PUSHR   #^M<R2,R4,R5>                           ; 0220
                    00C3   C9   95 00002                TSTB    195(IRAB)       ; 0289
                           07   13 00006                BEQL    1$
                50  867C   8F   3C 00008                MOVZWL  #34428, R0      ; 0291
                           46   11 0000D                BRB     6$
11        04    A9           15   E5 0000F 1$:         BBCC    #21, 4(IRAB), 2$ ; 0296
                54  0084   C9   D0 00014                MOVL    132(IRAB), BDB  ; 0298
                    0084   C9   D4 00019                CLRL    132(IRAB)
                           7E   D4 0001D                CLRL    -(SP)
                      0000G 30   00001F               BSBW    RM$RLSBKT
                5E         04   C0 00022               ADDL2   #4, SP
                54    20   A9   D0 00025 2$:          MOVL    32(IRAB), BDB   ; 0302
                55    18   A4   D0 00029              MOVL    24(BDB), BKT_ADDR ; 0303
                      0C   A5   95 0002D 3$:          TSTB    12(BKT_ADDR)    ; 0320
                           03   13 00030              BEQL    4$
                      0000G 30   00032                BSBW    RM$BUG3         ; 0321
                52    04   A5   3C 00035 4$:          MOVZWL  4(BKT_ADDR), END_OF_BUCKET ; 0326
                5?         55   C0 00039              ADDL2   BKT_ADDR, END_OF_BUCKET
```

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

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

```
;  345     0407  1 %SBTTL 'RM$PCKDEC_CHECK'
;  346     0408  1 GLOBAL ROUTINE RM$PCKDEC_CHECK : RL$RABREG_7 =
;  347     0409  1
;  348     0410  1 !++
;  349     0411  1 !
;  350     0412  1 !   FUNCTIONAL DESCRIPTION:
;  351     0413  1 !
;  352     0414  1 !       This routine verifies that a packed decimal key is valid. The key to
;  353     0415  1 !       be verified is always assumed to be in keybuffer 2.
;  354     0416  1 !
;  355     0417  1 !   CALLING SEQUENCE:
;  356     0418  1 !
;  357     0419  1 !       BSBW RM$PCKDEC_CHECK()
;  358     0420  1 !
;  359     0421  1 !   INPUT PARAMETERS:
;  360     0422  1 !       NONE
;  361     0423  1 !
;  362     0424  1 !   IMPLICIT INPUT:
;  363     0425  1 !
;  364     0426  1 !       IFAB                    - address of IFAB
;  365     0427  1 !           IFB$W_KBUFSZ        - size of one contigious keybuffer
;  366     0428  1 !
;  367     0429  1 !       IRAB                    - address of IRAB
;  368     0430  1 !           IRB$L_KEYBUF        - address of the contigious keybuffers
;  369     0431  1 !           IRB$B_KEYSZ         - size of key
;  370     0432  1 !
;  371     0433  1 !   OUTPUT PARAMETERS:
;  372     0434  1 !       NONE
;  373     0435  1 !
;  374     0436  1 !   IMPLICIT OUTPUT:
;  375     0437  1 !       NONE
;  376     0438  1 !
;  377     0439  1 !   ROUTINE_VALUE
;  378     0440  1 !
;  379     0441  1 !       SUC - string is a correct packed decimal string
;  380     0442  1 !       KEY - string is not a packed decimal string
;  381     0443  1 !
;  382     0444  1 !   SIDE EFFECTS
;  383     0445  1 !       NONE
;  384     0446  1 !
;  385     0447  1 !--
;  386     0448  1
;  387     0449  2     BEGIN
;  388     0450  2
;  389     0451  2     EXTERNAL REGISTER
;  390     0452  2         R_IFAB_STR,
;  391     0453  2         R_IRAB_STR;
;  392     0454  2
;  393     0455  2     LOCAL
;  394     0456  2         KEYBUF   : REF BLOCK [, BYTE];
;  395     0457  2
;  396     0458  2     MACRO
;  397     0459  2         LOW_MASK = 0,4,0 %,                     ! low nibble - digit (or sign)
;  398     0460  2         HIGH_MASK = 4,4,0 %;                    ! high nibble - digit
;  399     0461  2
;  400     0462  2     KEYBUF = KEYBUF_ADDR(2);
;  401     0463  2
```

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

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

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

```
;  426     0487  1 %SBTTL 'RM$PUT3B'
;  427     0488  1 GLOBAL ROUTINE RM$PUT3B : RL$RABREC =
;  428     0489  1
;  429     0490  1 !++
;  430     0491  1 !
;  431     0492  1 ! FUNCTIONAL DESCRIPTION:
;  432     0493  1 !
;  433     0494  1 !       The routine directs the insertion of a new record. The steps required
;  434     0495  1 !       to insert a new record are as follows:
;  435     0496  1 !
;  436     0497  1 !       1. Perform some operation specific validations and initializations.
;  437     0498  1 !
;  438     0499  1 !       2. Position to the point of insertion of the new record primary data
;  439     0500  1 !          record.
;  440     0501  1 !
;  441     0502  1 !       3. Convert the $PUT into an $UPDATE and perform the $UPDATE if duplicate
;  442     0503  1 !          primary keys are not allowed, the primary key of the new record is
;  443     0504  1 !          already represented within the file, update access is allowed, and
;  444     0505  1 !          updates have been requested in this particular circumstance.
;  445     0506  1 !
;  446     0507  1 !       4. Insert the new record into the current bucket provided there is both
;  447     0508  1 !          available record IDs and sufficient free space in the bucket to
;  448     0509  1 !          accomidate the new record. If there is not sufficient space or
;  449     0510  1 !          or record IDs, the primary data bucket is split, the new record is
;  450     0511  1 !          inserted where appropriate, and the primary index is updated as is
;  451     0512  1 !          necessary.
;  452     0513  1 !
;  453     0514  1 !       5. All alternate keys in the new record are inserted.
;  454     0515  1 !
;  455     0516  1 ! CALLING SEQUENCE:
;  456     0517  1 !
;  457     0518  1 !       BSBW RM$PUT3B()
;  458     0519  1 !
;  459     0520  1 ! INPUT PARAMETERS:
;  460     0521  1 !       NONE
;  461     0522  1 !
;  462     0523  1 ! IMPLICIT INPUTS:
;  463     0524  1 !
;  464     0525  1 !       IDX_DFN                 - index descriptor for primary key
;  465     0526  1 !           IDX$B_DATATYPE      - data type of primary key
;  466     0527  1 !           IDX$B_DATBKTYP      - primary data bucket compression type
;  467     0528  1 !           IDX$V_DUPKEYS       - if set, duplicate primary keys are allowed
;  468     0529  1 !           IDX$B_KEYSZ         - size of the primary key
;  469     0530  1 !
;  470     0531  1 !       IFAB                    - address of IFAB
;  471     0532  1 !           IFB$W_KBUFSZ        - size of a keybuffer
;  472     0533  1 !           IFB$B_PLG_VER       - prologue version of file
;  473     0534  1 !           IFB$B_RFMORG        - format of records
;  474     0535  1 !
;  475     0536  1 !       IRAB                    - address of IRAB
;  476     0537  1 !           IRB$V_CON_EOF       - if set, make sure record is at end-of-file
;  477     0538  1 !           IRB$L_CURBDB        - address of BDB for primary data bucket
;  478     0539  1 !           IRB$V_DUP           - if set, new record is a duplicate
;  479     0540  1 !           IRB$V_DUPS_SEEN     - if set, duplicate of new primary key seen
;  480     0541  1 !           IRB$L_KEYBOF        - address of te contigious keybuffers
;  481     0542  1 !           IRB$V_PUTS_LAST     - if set, last operation was sequential $PUT
;  482     0543  1 !           IRB$W_PUTUP_ID      - RFA ID of new primary data record
```

```
;  483     0544  1 !           IRB$L_PUTUP_VBN     - RFA VBN of new primary data record
;  484     0545  1 !           IRB$L_RBF           - address of user record buffer
;  485     0546  1 !           IRB$W_RSZ           - size of new record
;  486     0547  1 !           IRB$W_UDR_ID        - RFA ID of current primary data record
;  487     0548  1 !           IRB$L_UDR_VBN       - RFA VBN of current primary data record
;  488     0549  1 !           IRB$V_UNLOCK_RP     - if set, there is a current record lock
;  489     0550  1 !
;  490     0551  1 !       RAB                     - address of RAB
;  491     0552  1 !           RAB$B_RAC           - record access mode
;  492     0553  1 !
;  493     0554  1 ! OUTPUT PARAMETERS:
;  494     0555  1 !       NONE
;  495     0556  1 !
;  496     0557  1 ! IMPLICIT OUTPUTS:
;  497     0558  1 !
;  498     0559  1 !       IRAB
;  499     0560  1 !           IRB$V_CON_EOF       - 0
;  500     0561  1 !           IRB$V_IDX_ERR       - if set, error occurred updating index
;  501     0562  1 !           IRB$L_MIDX_TMP3     - VBN of insertion primary data bucket
;  502     0563  1 !           IRB$W_POS_INS       - offset of position of insertion
;  503     0564  1 !           IRB$V_POSINSERT     - 1
;  504     0565  1 !           IRB$V_PUTS_LAST     - if set, current operation is sequential $PUT
;  505     0566  1 !           IRB$W_PUTUP_ID      - RFA ID of new primary data record
;  506     0567  1 !           IRB$L_PUTUP_VBN     - RFA VBN of new primary data record
;  507     0568  1 !           IRB$V_UNLOCK_RP     - if set, new record is locked
;  508     0569  1 !
;  509     0570  1 !       RAB                     - address of RAB
;  510     0571  1 !           RAB$L_STV           - return value from RM$INSS_OR_IDX if error
;  511     0572  1 !                                 occurred updating index
;  512     0573  1 !
;  513     0574  1 ! ROUTINE VALUE:
;  514     0575  1 !
;  515     0576  1 !       RAC     - invalid record access mode
;  516     0577  1 !       SEQ     - new records not inserted in ascending sequential order
;  517     0578  1 !       SUC     - success
;  518     0579  1 !
;  519     0580  1 !       Also see return value from:
;  520     0581  1 !
;  521     0582  1 !       RM$CONFIRM_EOF
;  522     0583  1 !       RM$CSEARCH_TREE
;  523     0584  1 !       RM$INS_ALL_SIDR
;  524     0585  1 !       RM$LOCK
;  525     0586  1 !       RM$MAKE_INDEX
;  526     0587  1 !       RM$PCKDEC_CHECK
;  527     0588  1 !       RM$PUT_UPD_CHKS
;  528     0589  1 !       RM$PUT_UPD_FIN
;  529     0590  1 !       RM$PUT_UPD_SPL
;  530     0591  1 !       RM$RLSBKT
;  531     0592  1 !       RM$RU_JOURNAL3
;  532     0593  1 !       RM$UPDATE_IF
;  533     0594  1 !       RM$UPDATE3B
;  534     0595  1 !
;  535     0596  1 ! SIDE EFFECTS:
;  536     0597  1 !
;  537     0598  1 !       AP is trashed.
;  538     0599  1 !       Keybuffer 2 and 3 will have been trashed.
;  539     0600  1 !       If the current process is within a Recovery Unit, and the file is being
```

```
;  540     0601  1 !            Recovery Unit Journalled, then the operation is RU Journalled
;  541     0602  1 !            before any permanent modification to the file takes place.
;  542     0603  1 !        If the operation is a sequential $PUT, the keybuffer 6 will contain the
;  543     0604  1 !            new record.
;  544     0605  1 !        If the routine returns with success of any type then the new record has
;  545     0606  1 !            been inserted.
;  546     0607  1 !        If the routine returns with success then any required index updating has
;  547     0608  1 !            taken place.
;  548     0609  1 !        If the rotuine returns with a RVU error then the nwe record has been
;  549     0610  1 !            inserted and any required index updating has probably taken place,
;  550     0611  1 !            but some of the RRVs have probably not been correctly updated.
;  551     0612  1 !
;  552     0613  1 !--
;  553     0614  1
;  554     0615  2     BEGIN
;  555     0616  2
;  556     0617  2     EXTERNAL REGISTER
;  557     0618  2         COMMON_RAB_STR;
;  558     0619  2
;  559     0620  2     GLOBAL REGISTER
;  560     0621  2         R_BDB,
;  561     0622  2         R_IDX_DFN_STR,
;  562     0623  2         R_REC_ADDR_STR;
;  563     0624  2
;  564     0625  2     BUILTIN
;  565     0626  2         AP,
;  566     0627  2         TESTBITSC;
;  567     0628  2
;  568     0629  2     ! Do some validity checks and initialization, and unlock the current
;  569     0630  2     ! primary data record if there is one to unlock.
;  570     0631  2     !
;  571     0632  2     IRAB[IRB$V_UPDATE] = 0;
;  572     0633  2
;  573     0634  2     IF TESTBITSC(IRAB[IRB$V_UNLOCK_RP])
;  574     0635  2     THEN
;  575     0636  2         RM$UNLOCK (.IRAB[IRB$L_UDR_VBN], .IRAB[IRB$W_UDR_ID]);
;  576     0637  2
;  577     0638  2     RETURN_ON_ERROR (RM$PUT_UPD_CHKS());
;  578     0639  2
;  579     0640  2     ! Make sure the record access is valid - only sequential and key access
;  580     0641  2     ! is allowed.
;  581     0642  2     !
;  582     0643  2     ASSUME_C(RAB$C_KEY, 1);
;  583     0644  2
;  584     0645  2     IF .RAB[RAB$B_RAC] GTRU RAB$C_KEY
;  585     0646  2     THEN
;  586     0647  2         RETURN RMSERR(RAC);
;  587     0648  2
;  588     0649  2     ! If the record access mode is sequential and the last operation performed
;  589     0650  2     ! was also a sequential $PUT then make sure that the key of the new record
;  590     0651  2     ! is greater-than or equal to the key of the previous record inserted.
;  591     0652  2     !
;  592     0653  2     ! Move the key into keybuffer 6 and set the state bit IRB$V_PUTS_LAST for
;  593     0654  2     ! the next operation in case it too is a sequential $PUT.
;  594     0655  2     !
;  595     0656  2     IF .RAB[RAB$B_RAC] EQLU RAB$C_SEQ
;  596     0657  2     THEN
```

```
;  597     0658  3          BEGIN
;  598     0659  3
;  599     0660  3          IF .IRAB[IRB$V_PUTS_LAST]
;  600     0661  3          THEN
;  601     0662  4              BEGIN
;  602     0663  4
;  603     0664  4              AP = 2;
;  604     0665  4              IF  RM$COMPARE_KEY (.IRAB[IRB$L_RBF],
;  605     0666  4                                  KEYBUF_ADDR(6),
;  606     0667  4                                  .IDX_DFN[IDX$B_KEYSZ]) GTR 0
;  607     0668  4              THEN
;  608     0669  4                  RETURN RMSERR(SEQ);
;  609     0670  3              END;
;  610     0671  3
;  611     0672  3          AP = 3;
;  612     0673  3          REC_ADDR = .IRAB[IRB$L_RBF];
;  613     0674  3          RM$RECORD_KEY (KEYBUF_ADDR(6));
;  614     0675  3
;  615     0676  3          IRAB[IRB$V_PUTS_LAST] = 1;
;  616     0677  3          END
;  617     0678  2      ELSE
;  618     0679  2          IRAB[IRB$V_PUTS_LAST] = 0;
;  619     0680  2
;  620     0681  2      ! Prepare to position to the point of insertion of the new record within the
;  621     0682  2      ! primary index.
;  622     0683  2      !
;  623     0684  2      IRAB[IRB$L_CURBDB] = 0;
;  624     0685  2      IRAB[IRB$B_STOPLEVEL] = 0;
;  625     0686  2      IRAB[IRB$B_KEYSZ] = .IDX_DFN[IDX$B_KEYSZ];
;  626     0687  2      IRAB[IRB$W_SRCHFLAGS] = IRB$M_POSINSERT;
;  627     0688  2
;  628     0689  2      ! Move the record's key into keybuffer 2 and into keybuffer 3.
;  629     0690  2      !
;  630     0691  2      AP = 3;
;  631     0692  2      REC_ADDR = .IRAB[IRB$L_RBF];
;  632     0693  2      RM$RECORD_KEY (KEYBUF_ADDR(2));
;  633     0694  2
;  634     0695  2      CH$MOVE (.IDX_DFN[IDX$B_KEYSZ], KEYBUF_ADDR(2), KEYBUF_ADDR(3));
;  635     0696  2
;  636     0697  2      ! If the key is a packed decimal string, then call the packed decimal
;  637     0698  2      ! validation routine.
;  638     0699  2      !
;  639     0700  2      IF .IDX_DFN[IDX$B_DATATYPE] EQLU IDX$C_PACKED
;  640     0701  2      THEN
;  641     0702  2          RETURN_ON_ERROR (RM$PCKDEC_CHECK());
;  642     0703  2
;  643     0704  2      ! It is possible under a very restricted set of circumstances, that
;  644     0705  2      ! multiple positionings to the point of insertion will be required. Towards
;  645     0706  2      ! this end place the positioning code within an infinite loop that can only
;  646     0707  2      ! be exited when it is firmly decided what to do with the new record. The
;  647     0708  2      ! set of circumstances under which multiple positions will be required is:
;  648     0709  2      !
;  649     0710  2      ! 1. The primary key of the new record already exists in the file.
;  650     0711  2      ! 2. The file does not allow duplicate primary keys.
;  651     0712  2      ! 3. The user has allowed updates to the file
;  652     0713  2      ! 4. The user has requested an $UPDATE to be performed in such a case.
;  653     0714  2      ! 5. The stream must wait for the record lock on the record to be updated
```

```
;  654     0715  2      !    to be granted before performing the $UPDATE.
;  655     0716  2      !
;  656     0717  2      ! In such a situation RMS must re-position to the point of insertion to
;  657     0718  2      ! determine whether it should still perform an $UPDATE, or whether that
;  658     0719  2      ! record has been removed, and it need do only a $PUT of the new record.
;  659     0720  2      !
;  660     0721  2      WHILE 1
;  661     0722  2      DO
;  662     0723  3          BEGIN
;  663     0724  3
;  664     0725  3          LOCAL
;  665     0726  3              STATUS;
;  666     0727  3
;  667     0728  3          ! Position to the point of insertion of the new record.
;  668     0729  3          !
;  669     0730  3          IRAB[IRB$B_SPL_BITS] = 0;
;  670     0731  3          STATUS = RM$CSEARCH_TREE();
;  671     0732  3
;  672     0733  3          ! If no primary index exists then create one at this time and
;  673     0734  3          ! re-position.
;  674     0735  3          !
;  675     0736  4          IF .STATUS<0, 16> EQL RMSERR(IDX)
;  676     0737  3          THEN
;  677     0738  4              BEGIN
;  678     0739  4
;  679     0740  4              RETURN_ON_ERROR (RM$MAKE_INDEX());
;  680     0741  4
;  681     0742  4              STATUS = RM$CSEARCH_TREE();
;  682     0743  3              END;
;  683     0744  3
;  684     0745  3          ! Terminate the current operation if RMS could not position to the point
;  685     0746  3          ! of insertion of the new record.
;  686     0747  3          !
;  687     0748  3          IF NOT .STATUS
;  688     0749  3          THEN
;  689     0750  3              RETURN .STATUS;
;  690     0751  3
;  691     0752  3          ! If the IRB$V_CON_EOF state bit is set, verify that no record exists
;  692     0753  3          ! after the insert position which was just determined by
;  693     0754  3          ! RM$CSEARCH_TREE.  This will always be a best guess in a shared
;  694     0755  3          ! environment. This check is done by scanning the buckets along the
;  695     0756  3          ! horizontal chain looking for non-deleted records until the end of the
;  696     0757  3          ! chain is encountered. During this scan RMS must release the bucket
;  697     0758  3          ! containing the insertion point of the new record if it isn't the
;  698     0759  3          ! last bucket. This is because an RRV update could keep a rightward
;  699     0760  3          ! bucket locked and be trying to access this bucket. This would be a
;  700     0761  3          ! deadlock. If RMS is able to successful verify that there are no
;  701     0762  3          ! un-deleted records following the insertion point, then RMS
;  702     0763  3          ! re-positions to the point of insertion of the new record and
;  703     0764  3          ! continues.
;  704     0765  3          !
;  705     0766  3          ! Note: IRB$V_CON_EOF is turned off after the first $PUT or after any
;  706     0767  3          ! random $GET. Sequential $GETs will merely return RMS$_EOF until the
;  707     0768  3          ! IRB$V_CON_EOF bit is off.
;  708     0769  3          !
;  709     0770  3          IF .IRAB[IRB$V_CON_EOF]
;  710     0771  3          THEN
```

```
;  711     0772  4              BEGIN
;  712     0773  4
;  713     0774  4              ! Save the VBN of the bucket containing the insert position.
;  714     0775  4              ! RM$CONFIRM_EOF will use this VBN to re-position to this bucket
;  715     0776  4              ! prior to returning if and only if there are no non-deleted records
;  716     0777  4              ! following the current point of insertion.
;  717     0778  4              !
;  718     0779  4              IRAB[IRB$L_MIDX_TMP3] = .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$L_VBN];
;  719     0780  4
;  720     0781  4              ! Perform the check to see if there is a non-deleted record
;  721     0782  4              ! following the current position of insertion of the new record.
;  722     0783  4              ! If there is, return an error; otherwise, reposition to the
;  723     0784  4              ! point of insertion for the new record and clear the state bit.
;  724     0785  4              !
;  725     0786  4              RETURN_ON_ERROR (RM$CONFIRM_EOF());
;  726     0787  4              RETURN_ON_ERROR (RM$CSEARCH_TREE());
;  727     0788  4              IRAB[IRB$V_CON_EOF] = 0;
;  728     0789  3              END;
;  729     0790  3
;  730     0791  3          ! Make sure that RMS didn't see any duplicates if duplicates aren't
;  731     0792  3          ! allowed. If RMS did see them and they are allowed, set the IRB$V_DUP
;  732     0793  3          ! bit.
;  733     0794  3          !
;  734     0795  3          IF .IRAB[IRB$V_DUPS_SEEN]
;  735     0796  3          THEN
;  736     0797  3              IF NOT .IDX_DFN[IDX$V_DUPKEYS]
;  737     0798  3              THEN
;  738     0799  4                  BEGIN
;  739     0800  4
;  740     0801  4                  ! Determine whether an $UPDATE maybe done in place of the $PUT.
;  741     0802  4                  ! If so, set up do it, otherwise return the appropriate error.
;  742     0803  4                  !
;  743     0804  5                  IF NOT (STATUS = RM$UPDATE_IF())
;  744     0805  4                  THEN
;  745     0806  4
;  746     0807  4                      ! If there is already a record in the file with the same
;  747     0808  4                      ! primary key as the new record, but that record has been
;  748     0809  4                      ! RU_DELETEd by either this very same process or a process
;  749     0810  4                      ! that has successfully completed its recovery unit then
;  750     0811  4                      ! the new record is not a duplicate and a $PUT maybe done
;  751     0812  4                      ! to insert it. In this case the positioning loop maybe
;  752     0813  4                      ! left as the point of insertion of the new record has
;  753     0814  4                      ! been located.
;  754     0815  4                      !
;  755     0816  4                      IF .STATUS EQLU 0
;  756     0817  4                      THEN
;  757     0818  4                          EXITLOOP
;  758     0819  4
;  759     0820  4                      ! Otherwise, some error has been encountered, so return
;  760     0821  4                      ! the appropriate status.
;  761     0822  4                      !
;  762     0823  4                      ELSE
;  763     0824  4                          RETURN .STATUS;
;  764     0825  4
;  765     0826  4                  ! If RMS was able to lock the record to be updated without any
;  766     0827  4                  ! difficulties, and furthermore, found that it had not been
;  767     0828  4                  ! deleted within any recovery unit by this or any other process,
```

```
;  768     0829  4              ! then go ahead and update it.
;  769     0830  4              !
;  770     0831  5              IF .STATUS<0,16> NEQU RMSSUC(OK_WAT)
;  771     0832  4              THEN
;  772     0833  5                  BEGIN
;  773     0834  5                  IRAB[IRB$V_UPDATE_IF] = 1;
;  774     0835  5                  STATUS = RM$UPDATE3B();
;  775     0836  5                  IRAB[IRB$V_UPDATE_IF] = 0;
;  776     0837  5                  RETURN .STATUS;
;  777     0838  4                  END;
;  778     0839  4
;  779     0840  4              ! If RMS had to wait for a record lock to be granted on the
;  780     0841  4              ! record which is to be updated, then while it waited it
;  781     0842  4              ! released its lock on the primary data bucket. RMS must now
;  782     0843  4              ! re-position back to the record, and the easiest way to do
;  783     0844  4              ! this (taking into account that the record might have been
;  784     0845  4              ! deleted and RMS would want to do a $PUT now) is to go through
;  785     0846  4              ! the re-positioning loop one more time. The lock on the record
;  786     0847  4              ! itself has already been released.
;  787     0848  4              !
;  788     0849  4              END
;  789     0850  4
;  790     0851  4          ! The positioning loop maybe left because the new record's primary
;  791     0852  4          ! key has been found to be a duplicate, duplicates are allowed, and
;  792     0853  4          ! thus the point of insertion of the new record has been located.
;  793     0854  4          !
;  794     0855  3          ELSE
;  795     0856  4              BEGIN
;  796     0857  4              IRAB[IRB$V_DUP] = 1;
;  797     0858  4              EXITLOOP;
;  798     0859  4              END
;  799     0860  4
;  800     0861  4              ! The positioning loop maybe left because the new record's primary key
;  801     0862  4              ! is not represented in the file, and thus, the point of insertion of
;  802     0863  4              ! the new record has been located.
;  803     0864  4              !
;  804     0865  3              ELSE
;  805     0866  3                  EXITLOOP;
;  806     0867  3
;  807     0868  2              END;
;  808     0869  2
;  809     0870  2          ! Prepare to insert the primary data record.
;  810     0871  2          !
;  811     0872  3          BEGIN
;  812     0873  3
;  813     0874  3          GLOBAL REGISTER
;  814     0875  3              COMMON_IO_STR;
;  815     0876  3
;  816     0877  3          BDB = .IRAB[IRB$L_CURBDB];
;  817     0878  3          BKT_ADDR = .BDB[BDB$L_ADDR];
;  818     0879  3
;  819     0880  3          ! Insert the primary data record. To do so the size of the primary data
;  820     0881  3          ! record is computed, and an attempt is made to insert it into the current
;  821     0882  3          ! primary data bucket at the point of insertion. If there is insufficient
;  822     0883  3          ! freespace in the bucket or insufficient record IDs, then a primary data
;  823     0884  3          ! bucket split is done, and the new record is inserted appropriately.
;  824     0885  3          !
```

```
;  825     0886  4     BEGIN
;  826     0887  4
;  827     0888  4     LOCAL
;  828     0889  4         ID_SPLIT,
;  829     0890  4         RECORD_SIZE;
;  830     0891  4
;  831     0892  4     ID_SPLIT = 0;
;  832     0893  4
;  833     0894  4     ! Determine the amount of space the new record will occupy in the bucket,
;  834     0895  4     ! and check for the availability of record IDs. If there are no record IDs
;  835     0896  4     ! available in the bucket then the primary data bucket will have to be
;  836     0897  4     ! split regardless of whether or not there is sufficient space in the
;  837     0898  4     ! bucket for the new record. Both of these determinations are prologue
;  838     0899  4     ! dependent.
;  839     0900  4     !
;  840     0901  4     IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  841     0902  4     THEN
;  842     0903  5         BEGIN
;  843     0904  5         RECORD_SIZE = .IRAB[IRB$W_RSZ] + IRC$C_FIXOVHDSZ;
;  844     0905  5
;  845     0906  5         IF .IFAB[IFB$B_RFMORG] NEQU FAB$C_FIX
;  846     0907  5         THEN
;  847     0908  5             RECORD_SIZE = .RECORD_SIZE + IRC$C_DATSZFLD;
;  848     0909  5
;  849     0910  5         ! Check for the availability of record IDs. Because the file is a
;  850     0911  5         ! prologue 1 or 2 file there maybe some free record IDs within the
;  851     0912  5         ! the bucket. Free IDs occur when records are moved out of a bucket as
;  852     0913  5         ! the result of a bucket split and the primary data bucket splitting
;  853     0914  5         ! was not their original bucket. If free IDs are found, they maybe used.
;  854     0915  5         !
;  855     0916  5         IF .BKT_ADDR[BKT$B_NXTRECID] EQLU 0
;  856     0917  5              OR
;  857     0918  5            .BKT_ADDR[BKT$B_NXTRECID] GTRU .BKT_ADDR[BKT$B_LSTRECID]
;  858     0919  5         THEN
;  859     0920  5             IF NOT RM$BKT_SORT(.BKT_ADDR)
;  860     0921  5             THEN
;  861     0922  6                 BEGIN
;  862     0923  6                 ID_SPLIT = 1;
;  863     0924  6                 BKT_ADDR[BKT$B_NXTRECID] = 0;
;  864     0925  6                 IRAB[IRB$W_POS_INS] = .REC_ADDR - .BKT_ADDR;
;  865     0926  5                 END;
;  866     0927  5         END
;  867     0928  4     ELSE
;  868     0929  5         BEGIN
;  869     0930  5         RECORD_SIZE = RM$PACK_REC();
;  870     0931  5         RECORD_SIZE = .RECORD_SIZE + IRC$C_FIXOVHSZ3;
;  871     0932  5
;  872     0933  5         ! All version 3 files have a record size, except fixed length records
;  873     0934  5         ! that have not been compressed.
;  874     0935  5         !
;  875     0936  5         IF .IFAB[IFB$B_RFMORG] NEQU FAB$C_FIX
;  876     0937  5             OR
;  877     0938  6            (.IFAB[IFB$B_RFMORG] EQL FAB$C_FIX
;  878     0939  6                         AND
;  879     0940  6                        .IDX_DFN[IDX$B_DATBKTYP] NEQU IDX$C_NCMPNCMP)
;  880     0941  5         THEN
;  881     0942  5             RECORD_SIZE = .RECORD_SIZE + IRC$C_DATSZFLD;
```

```
;  882          0943  5
;  883          0944  5          ! Check for the availability of record IDs.
;  884          0945  5          !
;  885          0946  5          IF .BKT_ADDR[BKT$W_NXTRECID] EQLU 0
;  886          0947  5          THEN
;  887          0948  6              BEGIN
;  888          0949  6              ID_SPLIT = 1;
;  889          0950  6              IRAB[IRB$W_POS_INS] = .REC_ADDR - .BKT_ADDR;
;  890          0951  5              END;
;  891          0952  4          END;
;  892          0953  4          !
;  893          0954  4      ! If both record IDs and sufficient space are available in the bucket then
;  894          0955  4      ! insert the new record; otherwise, split the primary data record inserting
;  895          0956  4      ! the new record in the process.
;  896          0957  4      !
;  897          0958  4      IF   .ID_SPLIT
;  898          0959  4           OR
;  899          0960  4           NOT RM$INSERT_UDR (RECORD_SIZE)
;  900          0961  4      THEN
;  901        P 0962  4          RETURN_ON_ERROR (RM$PUT_UPD_SPL (.RECORD_SIZE), BEGIN
;  902        P 0963  4                                                          IRAB[IRB$V_UPDATE] = 0;
;  903        P 0964  4                                                          RM$CLEAN_BDB()
;  904          0965  5                                                          END)
;  905          0966  5          !
;  906          0967  5      ! If RMS was able to insert the new record into the current bucket without
;  907          0968  5      ! splitting it and if the operation is taking place on a RU Journallable
;  908          0969  5      ! file within a Recovery Unit, then RU Journal the operation at this time
;  909          0970  5      ! before the modified primary data bucket is released.
;  910          0971  5      !
;  911          0972  5      ! NOTE: if a primary bucket data split was required, then the RU
;  912          0973  5      !       Journalling of this operation would have taken place within
;  913          0974  5      !       RM$PUT_UPD_SPL before the modified original bucket was released.
;  914          0975  5      !
;  915          0976  4      ELSE
;  916          0977  4          IF .IFAB[IFB$V_RUP]
;  917          0978  4          THEN
;  918          0979  5              BEGIN
;  919          0980  5              REC_ADDR = .IRAB[IRB$L_RBF];
;  920          0981  5              !
;  921        P 0982  5              RETURN_ON_ERROR (RM$RU_JOURNAL3 (RJR$_PUT,
;  922        P 0983  5                                               .IRAB[IRB$L_PUTUP_VBN],
;  923        P 0984  5                                               .IRAB[IRB$W_PUTUP_ID],
;  924        P 0985  5                                               .IRAB[IRB$W_RSZ]),
;  925          0986  5                               RM$CLEAN_BDB());
;  926          0987  4              END;
;  927          0988  3      END;
;  928          0989  3      !
;  929          0990  3      ! The new record, having been inserted, is now locked.
;  930          0991  3      !
;  931        P 0992  3      RETURN_ON_ERROR (RM$LOCK (.IRAB[IRB$L_PUTUP_VBN], .IRAB[IRB$W_PUTUP_ID]),
;  932        P 0993  3          BEGIN
;  933        P 0994  3          IRAB[IRB$V_UPDATE] = 0;
;  934        P 0995  3          RM$CLEAN_BDB()
;  935          0996  3          END);
;  936          0997  3      !
;  937          0998  3      IF NOT .RAB[RAB$V_ULK]
;  938          0999  3      THEN
```

```
;  939     1000  3          IRAB[IRB$V_UNLOCK_RP] = 1;
;  940     1001  3
;  941     1002  3      ! If the primary data bucket has split, and the primary index must be
;  942     1003  3      ! updated, then do so at this time.
;  943     1004  3      !
;  944     1005  3      IF TESTBITSC (IRAB[IRB$V_UPDATE])
;  945     1006  3      THEN
;  946     1007  4          BEGIN
;  947     1008  4
;  948     1009  4          LOCAL
;  949     1010  4              STATUS;
;  950     1011  4
;  951     1012  4          IRAB[IRB$B_STOPLEVEL] = 1;
;  952     1013  4          IRAB[IRB$W_SRCHFLAGS] = IRB$M_POSINSERT;
;  953     1014  4
;  954     1015  4          ! If an error occurs and no previous error has been saved, save
;  955     1016  4          ! the new error status.
;  956     1017  4          !
;  957     1018  5          IF NOT (STATUS = RM$INSS_OR_IDX())
;  958     1019  4          THEN
;  959     1020  5              BEGIN
;  960     1021  5              IF .RAB[RAB$L_STV] EQL 0 THEN
;  961     1022  5                  RAB[RAB$L_STV] = .STATUS OR 1^16; ! 1^16 is RMS's facility code
;  962     1023  5              IRAB[IRB$V_IDX_ERR] = 1;
;  963     1024  5              END
;  964     1025  5
;  965     1026  4          END
;  966     1027  4
;  967     1028  4      ! If no index update is required, then release the primary data bucket
;  968     1029  4      ! and any level above that might have been locked during positioning.
;  969     1030  4      !
;  970     1031  3      ELSE
;  971     1032  4          BEGIN
;  972     1033  4
;  973     1034  4          ! Note that the original bucket may have been written with a keep lock
;  974     1035  4          ! or may not have been released at all yet, depending on the path taken
;  975     1036  4          ! to get here. This is assuming that we do not want to force a write of
;  976     1037  4          ! the bucket if no split has taken place to take advantage of deferred
;  977     1038  4          ! write. If this notion changes, so must this code.
;  978     1039  4          !
;  979     1040  4          BDB = .IRAB[IRB$L_CURBDB];
;  980     1041  4          IRAB[IRB$L_CURBDB] = 0;
;  981     1042  4
;  982     1043  4          RETURN_ON_ERROR (RM$RLSBKT(0), RM$CLEAN_BDB());
;  983     1044  4
;  984     1045  4          IF (BDB = .IRAB[IRB$L_LOCK_BDB]) NEQ 0
;  985     1046  4          THEN
;  986     1047  5              BEGIN
;  987     1048  5              RM$RLSBKT(0);
;  988     1049  5              IRAB[IRB$L_LOCK_BDB] = 0;
;  989     1050  4              END;
;  990     1051  3          END;
;  991     1052  3
;  992     1053  3      ! Insert all of the alternate keys.
;  993     1054  3      !
;  994     1055  3      RETURN_ON_ERROR (RM$INS_ALL_SIDR(), RM$CLEAN_BDB());
;  995     1056  3
```

```
;  996     1057  3     ! Finish up the operation.
;  997     1058  3     !
;  998     1059  3     RETURN RM$PUT_UPD_FIN();
;  999     1060  3
; 1000     1061  3     END
; 1001     1062  1     END;


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

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

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

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

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

; 1002          1063  1
```

```
;  1004          1064  1 %SBTTL 'RM$UPDATE_IF'
;  1005          1065  1 ROUTINE RM$UPDATE_IF : RL$LINKAGE =
;  1006          1066  1
;  1007          1067  1 !++
;  1008          1068  1 !
;  1009          1069  1 ! FUNCTIONAL DESCRIPTION:
;  1010          1070  1 !
;  1011          1071  1 !       Perform all of the pre-processing necessary to convert a $PUT into
;  1012          1072  1 !       an $UPDATE. This includes:
;  1013          1073  1 !
;  1014          1074  1 !       1. Positioning to the record to be updated.
;  1015          1075  1 !       2. Locking the primary data record provided record locking has not been
;  1016          1076  1 !          disabled.
;  1017          1077  1 !       3. Determining whether this record can be updated, or whether the new
;  1018          1078  1 !          record should still be inserted based upon:
;  1019          1079  1 !
;  1020          1080  1 !          a. The ease with which the record lock was obtained.
;  1021          1081  1 !          b. The RU_DELETE status of the record which has just been locked.
;  1022          1082  1 !          c. Whether updates are allowed at all on the file.
;  1023          1083  1 !          d. Whether updates are allowed in this specific case.
;  1024          1084  1 !
;  1025          1085  1 ! CALLING SEQUENCE:
;  1026          1086  1 !
;  1027          1087  1 !       BSBW RM$UPDATE_IF()
;  1028          1088  1 !
;  1029          1089  1 ! INPUT PARAMETERS:
;  1030          1090  1 !       NONE
;  1031          1091  1 !
;  1032          1092  1 ! IMPLICIT INPUTS:
;  1033          1093  1 !
;  1034          1094  1 !       IFAB                    - address of IFAB
;  1035          1095  1 !           IFB$V_NORECLK       - if set, record locking is disabled
;  1036          1096  1 !           IFB$V_UPD           - if set, updates are allowed on this file
;  1037          1097  1 !           IFB$B_RUP           - if set Recovery Unit is in progress
;  1038          1098  1 !           IFB$V_RU_RLK        - if set, perform pseudo record locking
;  1039          1099  1 !
;  1040          1100  1 !       IRAB                    - address of IRAB
;  1041          1101  1 !           IRB$L_CURBDB        - address of buffer descriptor for UDR's bucket
;  1042          1102  1 !           IRB$L_LST_REC       - address of last non-deleted record
;  1043          1103  1 !
;  1044          1104  1 !       RAB                     - address of user RAB control block
;  1045          1105  1 !           RAB$V_UIF           - if set, a $PUT maybe converted into $UPDATE
;  1046          1106  1 !
;  1047          1107  1 !       REC_ADDR                - address following record to be updated
;  1048          1108  1 !
;  1049          1109  1 ! OUTPUT PARAMETERS:
;  1050          1110  1 !       NONE
;  1051          1111  1 !
;  1052          1112  1 ! IMPLICIT OUTPUTS:
;  1053          1113  1 !
;  1054          1114  1 !       IRAB                    - address of IRAB
;  1055          1115  1 !           IRB$W_PUTUPD_ID     - ID of primary data record to be updated
;  1056          1116  1 !           IRB$L_PUTUPD_VBN    - VBN of primary data record to be updated
;  1057          1117  1 !           IRB$V_UNLOCK_RP     - if set, release current record lock on errors
;  1058          1118  1 !
;  1059          1119  1 ! ROUTINE VALUE:
;  1060          1120  1 !
```

```
;  1061     1121  1 !       0       - a $PUT is to be performed
;  1062     1122  1 !       RLK     - record to be updated is locked by another stream
;  1063     1123  1 !       DME     - dynamic memory exhausted
;  1064     1124  1 !       SUC     - success
;  1065     1125  1 !       OK_WAT  - success, but had to wait for record lock
;  1066     1126  1 !       OK_ALK  - success, but this stream already had this record locked
;  1067     1127  1 !       OK_RULK - success, but this stream had locked this record within a RU
;  1068     1128  1 !
;  1069     1129  1 ! SIDE EFFECTS:
;  1070     1130  1 !
;  1071     1131  1 !       If a decision is made to perform an $UPDATE, then the record to be
;  1072     1132  1 !           updated is positioned to, made the current record, and locked
;  1073     1133  1 !           (the latter if record locking is not disabled). Under certain
;  1074     1134  1 !           circumstances, this record maybe reformated.
;  1075     1135  1 !       If a decision is made to perform a $PUT, then any record locks requested
;  1076     1136  1 !           within this routine are released and this routine returns 0 with
;  1077     1137  1 !           REC_ADDR pointing at where the new record is to be inserted. Under
;  1078     1138  1 !           certain circumstances a record marked RU_DELETE might have been
;  1079     1139  1 !           deleted from the file.
;  1080     1140  1 !       On any errors, or when RMS decides it must perform a re-positioning,
;  1081     1141  1 !           any record locks granted within this routine are released.
;  1082     1142  1 !       On errors that terminate the current operation, this routine releases
;  1083     1143  1 !           all bucket locks held by the stream.
;  1084     1144  1 !
;  1085     1145  1 !--
;  1086     1146  1
;  1087     1147  2     BEGIN
;  1088     1148  2
;  1089     1149  2     BUILTIN
;  1090     1150  2         AP,
;  1091     1151  2         TESTBITSC;
;  1092     1152  2
;  1093     1153  2     EXTERNAL REGISTER
;  1094     1154  2         COMMON_RAB_STR,
;  1095     1155  2         R_IDX_DFN_STR,
;  1096     1156  2         R_REC_ADDR_STR;
;  1097     1157  2
;  1098     1158  2     LOCAL
;  1099     1159  2         ID,
;  1100     1160  2         SAVE_REC_ADDR,
;  1101     1161  2         STATUS,
;  1102     1162  2         VBN;
;  1103     1163  2
;  1104     1164  2     ! Position to the preceeding record, the record which is to be updated.
;  1105     1165  2     !
;  1106     1166  2     SAVE_REC_ADDR = .REC_ADDR;
;  1107     1167  2     REC_ADDR = .IRAB[IRB$L_LST_REC];
;  1108     1168  2
;  1109     1169  2     ! If record locking is enabled, lock the record RMS has positioned to,
;  1110     1170  2     ! otherwise, assume success for the time being.
;  1111     1171  2     !
;  1112     1172  2     AP = 3;
;  1113     1173  2
;  1114     1174  3     BEGIN
;  1115     1175  3
;  1116     1176  3     GLOBAL REGISTER
;  1117     1177  3         R_BDB;
```

```
;  1118          1178  3
;  1119          1179  3      VBN = RMSRECORD_VBN();
;  1120          1180  2      END;
;  1121          1181  2
;  1122          1182  2      ID  = RMSRECORD_ID();
;  1123          1183  2
;  1124          1184  2      IF  NOT .IFAB[IFBSV_NORECLK]
;  1125          1185  2           OR
;  1126          1186  3          (.IFAB[IFBSV_RU_RLK]
;  1127          1187  3                    AND
;  1128          1188  3                   .IFAB[IFBSV_RUP])
;  1129          1189  2      THEN
;  1130          1190  3          IF (STATUS = RMSLOCK (.VBN, .ID))
;  1131          1191  2              AND
;  1132          1192  3              .STATUS<0,16> NEQU RMSSUC(OK_ALK)
;  1133          1193  2          THEN
;  1134          1194  2              IRAB[IRBSV_UNLOCK_RP] = 1
;  1135          1195  2          ELSE
;  1136          1196  2              IRAB[IRBSV_UNLOCK_RP] = 0
;  1137          1197  2      ELSE
;  1138          1198  2          STATUS = RMSSUC();
;  1139          1199  2
;  1140          1200  2      ! RMS is able to lock the record it has positioned to without waiting for
;  1141          1201  2      ! the lock to be granted, but finds that this record has been deleted within
;  1142          1202  2      ! a recovery unit. In this case, RMS may proceed to insert the new record
;  1143          1203  2      ! instead of converting the current operation into an $UPDATE. Furthermore,
;  1144          1204  2      ! if the RU deleted record had been deleted within another process's
;  1145          1205  2      ! recovery unit (which has since completed successfully), RMS may now delete
;  1146          1206  2      ! this record.
;  1147          1207  2      !
;  1148          1208  2      IF  .STATUS
;  1149          1209  2           AND
;  1150          1210  3          .STATUS<0,16> NEQU RMSSUC(OK_WAT)
;  1151          1211  2           AND
;  1152          1212  2          .REC_ADDR[IRCSV_RU_DELETE]
;  1153          1213  2      THEN
;  1154          1214  3          BEGIN
;  1155          1215  3
;  1156          1216  3          ! Unless this stream already had this record locked, release the lock
;  1157          1217  3          ! that was granted above. In either case, clear the bit IRBSV_UNLOCK_RP
;  1158          1218  3          ! so no attempt will be made to release a record lock on any subsequent
;  1159          1219  3          ! errors.
;  1160          1220  3          !
;  1161          1221  3          IF TESTBITSC (IRAB[IRBSV_UNLOCK_RP])
;  1162          1222  3          THEN
;  1163          1223  3              RMSUNLOCK (.VBN, .ID);
;  1164          1224  3
;  1165          1225  3          ! If the RU deleted record has been deleted within another process's
;  1166          1226  3          ! recovery unit or within the current process's recovery unit and the
;  1167          1227  3          ! recovery unit has successfully completed, then delete the record for
;  1168          1228  3          ! good, and re-position to the new record's point of insertion.
;  1169          1229  3          !
;  1170          1230  5          IF NOT ((.STATUS<0,16> EQLU RMSSUC(OK_RULK))
;  1171          1231  4                  OR
;  1172          1232  6                 (.STATUS<0,16> EQLU RMSSUC(OK_ALK)
;  1173          1233  5                           AND
;  1174          1234  4                          .IFAB[IFBSV_RUP]))
```

```
;1175     1235  3          THEN
;1176     1236  4              BEGIN
;1177     1237  4
;1178     1238  4              GLOBAL REGISTER
;1179     1239  4                  R_BDB_STR,
;1180     1240  4                  R_BKT_ADDR;
;1181     1241  4
;1182     1242  4              LOCAL
;1183     1243  4                  SAVE_REC_ID;
;1184     1244  4
;1185     1245  4              SAVE_REC_ID = .REC_ADDR[IRC$W_ID];
;1186     1246  4              REC_ADDR[IRC$V_RU_DELETE] = 0;
;1187     1247  4              REC_ADDR[IRC$V_RU_UPDATE] = 0;
;1188     1248  4
;1189     1249  4              BDB = .IRAB[IRB$L_CURBDB];
;1190     1250  4              RM$DELETE_UDR();
;1191     1251  4              BDB[BDB$V_DRT] = 1;
;1192     1252  4
;1193     1253  4              ! If the ID of the record REC_ADDR points to is the same as the
;1194     1254  4              ! ID of the record just deleted, then the record could not be
;1195     1255  4              ! completed deleted; therefore, position past it to the point of
;1196     1256  4              ! insertion of the new record.
;1197     1257  4              !
;1198     1258  4              IF .SAVE_REC_ID EQLU .REC_ADDR[IRC$W_ID]
;1199     1259  4              THEN
;1200     1260  4                  RM$GETNEXT_REC();
;1201     1261  4
;1202     1262  4              END
;1203     1263  4
;1204     1264  4          ! If the record which has been locked was already locked by this process
;1205     1265  4          ! (presumably within a recovery unit), then this record can not be
;1206     1266  4          ! deleted at this time. Just restore the address of the point of
;1207     1267  4          ! insertion of the new record.
;1208     1268  4          !
;1209     1269  3          ELSE
;1210     1270  3              REC_ADDR = .SAVE_REC_ADDR;
;1211     1271  3
;1212     1272  3          ! Return a status of 0 to signal that a $PUT should still be done.
;1213     1273  3          !
;1214     1274  3          STATUS = 0;
;1215     1275  3          END
;1216     1276  3
;1217     1277  3      ! RMS is able to lock the record it has positioned to without waiting for
;1218     1278  3      ! the lock to be granted, and furthermore finds that this record has not
;1219     1279  3      ! been deleted within a recovery unit; or, it encountered an error in its
;1220     1280  3      ! attempt to lock the record. In the former case, RMS may proceed to $UPDATE
;1221     1281  3      ! this record provided updates are allowed in this particular circumstance.
;1222     1282  3      ! In the latter case, RMS will change the error to something more
;1223     1283  3      ! appropriate if the record had been locked and updates would not have been
;1224     1284  3      ! allowed. If this routine is to return an error, all locked buckets are
;1225     1285  3      ! released at this point.
;1226     1286  3      !
;1227     1287  2      ELSE
;1228     1288  3          IF .STATUS<0,16> NEQU RMSSUC(OK_WAT)
;1229     1289  2          THEN
;1230     1290  3              BEGIN
;1231     1291  3
```

```
;  1232     1292  3            ! Map the current status into the appropriate error status if the
;  1233     1293  3            ! file has not been opened for update access or updates are not
;  1234     1294  3            ! allowed in this particular circumstance (ie - a $PUT may not be
;  1235     1295  3            ! converted into an $UPDATE).
;  1236     1296  3            !
;  1237     1297  3            IF NOT .RAB[RAB$V_UIF]
;  1238     1298  3            THEN
;  1239     1299  4                STATUS = RMSERR(DUP)
;  1240     1300  3            ELSE
;  1241     1301  3                IF NOT .IFAB[IFB$V_UPD]
;  1242     1302  3                THEN
;  1243     1303  4                    STATUS = RMSERR(FAC);
;  1244     1304  3
;  1245     1305  3            ! If a $PUT maybe converted into an $UPDATE, then setup to do the
;  1246     1306  3            ! $UPDATE.
;  1247     1307  3            !
;  1248     1308  3            IF .STATUS
;  1249     1309  3            THEN
;  1250     1310  4                BEGIN
;  1251     1311  4
;  1252     1312  4                IRAB[IRB$L_PUTUP_VBN] = .VBN;
;  1253     1313  4                IRAB[IRB$W_PUTUP_ID]  = .ID;
;  1254     1314  4
;  1255     1315  4                ! Under the following set of circumstances, reformat the record
;  1256     1316  4                ! which is to be updated:
;  1257     1317  4                !
;  1258     1318  4                ! 1. The record was put into a special format after a previous
;  1259     1319  4                !    $UPDATEing within a recove y unit in order to preserve
;  1260     1320  4                !    space for t.e duration of .he recovery unit which otherwise
;  1261     1321  4                !    would have been lost.
;  1262     1322  4                ! 2. Either the current proce's is not in a recovery unit, or
;  1263     1323  4                !    the current stream was a le to directly obtain the record
;  1264     1324  4                !    indicating the completi n of the recovery unit in which
;  1265     1325  4                !    the previous $UPDATE ha  been done.
;  1266     1326  4                !
;  1267     1327  4                IF .REC_ADDR[IRC$V_RU_UPDAT ]
;  1268     1328  4                THEN
;  1269     1329  4                    IF  NOT .IFAB[IFB$V  JP]
;  1270     1330  4                         OR
;  1271     1331  5                        .STATUS<0,16> _QLU RMSSUC()
;  1272     1332  4                    THEN
;  1273     1333  5                        BEGIN
;  1274     1334  5
;  1275     1335  5                        GLOBA_ REGISTER
;  1276     1336  5                             _BKT_ADDR;
;  1277     1337  5
;  1278     1338  5                        BK_ ADDR = .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$L_ADDR];
;  1279     1339  5                        RM_RU_REFORMAT();
;  1280     1340  4                        E'D;
;  1281     1341  4                END
;  1282     1342  4
;  1283     1343  4            ! If some error has been encountered in this routine whether in
;  1284     1344  4            ! locking the record to be updated or in the conversion of the $PUT
;  1285     1345  4            ! into t.e $UPDATE, then release all bucket locks and any record
;  1286     1346  4            ! locks that this routine has obtained before returning the error.
;  1287     1347  4            !
;  1288     1348  3            ELSE
```

```
;  1289          1349  4                  BEGIN
;  1290          1350  4
;  1291          1351  4                  IF TESTBITSC (IRAB[IRB$V_UNLOCK_RP])
;  1292          1352  4                  THEN
;  1293          1353  4                      RM$UNLOCK (.VBN, .ID);
;  1294          1354  4
;  1295          1355  4                  RM$CLEAN_BDB();
;  1296          1356  3                  END;
;  1297          1357  3
;  1298          1358  3              END
;  1299          1359  3
;  1300          1360  3          ! If RMS had to wait for a record lock to be granted, then while it
;  1301          1361  3          ! was waiting it had to release the bucket containing the record it
;  1302          1362  3          ! eventually was able to lock. Because anything might have happened
;  1303          1363  3          ! once this stream released the primary data bucket including the
;  1304          1364  3          ! deletion of the record RMS has locked, what RMS must do is force
;  1305          1365  3          ! a re-positioning so that it may re-determine whether it should
;  1306          1366  3          ! insert the new record or use it to update an existing record.
;  1307          1367  3          ! Before returning so that the re-positioning maybe done, RMS releases
;  1308          1368  3          ! the record lock it had to wait for.
;  1309          1369  3          !
;  1310          1370  2          ELSE
;  1311          1371  3              BEGIN
;  1312          1372  3              IRAB[IRB$V_UNLOCK_RP] = 0;
;  1313          1373  3
;  1314          1374  4              IF (STATUS = RM$UNLOCK (.VBN, .ID))
;  1315          1375  3              THEN
;  1316          1376  4                  STATUS = RMSSUC(OK_WAT)
;  1317          1377  3              ELSE
;  1318          1378  3                  RM$CLEAN_BDB();
;  1319          1379  2              END;
;  1320          1380  2
;  1321          1381  2      ! Return status.
;  1322          1382  2      !
;  1323          1383  2      RETURN .STATUS;
;  1324          1384  2
;  1325          1385  1      END;


                                    3C  BB 00000 RM$UPDATE_IF:
                                                 PUSHR   #^M<R2,R3,R4,R5>                    ; 1065
                              5E            04  C2 00002                SUBL2   #4, SP                              ;
                              55            56  D0 00005                MOVL    REC_ADDR, SAVE_REC_ADDR             ; 1166
                              56       4C   A9  D0 00008                MOVL    76(IRAB), REC_ADDR                  ; 1167
                              5C            03  D0 0000C                MOVL    #3, AP                              ; 1172
                                       0000G 30  0000F                BSBW    RM$RECORD_VBN                       ; 1179
                              54            50  D0 00012                MOVL    R0, VBN                             ;
                                       0000G 30  00015                BSBW    RM$RECORD_ID                        ; 1182
                              52            50  D0 00018                MOVL    R0, ID                              ;
                        0C       06   AA        03  E1 0001B           BBC     #3, 6(IFAB), 1$                     ; 1184
                        28     00A2   CA        03  F1 00020           BBC     #3, 162(IFAB), 3$                   ; 1186
                        22     00A2   CA        02  E1 00026           BBC     #2, 162(IFAB), 3$                   ; 1188
                              51            54  D0 0002C 1$:           MOVL    VBN, R1                             ; 1190
                                 00000000G  EF  16 0002F                JSB     RM$LOCK                             ;
```

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

```
                    00000000G  EF  16 00107          JSB     RM$UNLOCK                                       ;
                 6E            50  D0 0010D          MOVL    R0, STATUS                                      ;
                 07            6E  E9 00110          BLBC    STATUS, 15$                                     ;
                 6E      8061  8F  3C 00113          MOVZWL  #32865, STATUS                                  ; 1376
                               03  11 00118          BRB     16$                                             ;
                         0000G 30  00119 15$:        BSBW    RM$CLEAN_BDB                                    ; 1378
                 50            8E  D0 0011D 16$:     MOVL    STATUS, R0                                      ; 1383
                               3C  BA 00120          POPR    #^M<R2,R3,R4,R5>                                ; 1385
                                   05 00122          RSB                                                     ;

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

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



;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  RM$RMS3                      1184  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  GBL,  REL,  CON,  PIC,ALIGN(2)



;                                 Library Statistics
;
;                                          --------- Symbols ---------      Pages       Processing
;       File                               Total     Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[RMS.OBJ]RMS.L32;1           3109        123         3       154         00:00.4



;                                       COMMAND QUALIFIERS
;
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3PUT/OBJ=OBJ$:RM3PUT MSRC$:RM3PUT/UPDATE=(ENH$:RM3PUT)

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

RM3PROBE
LIS

RM3SIOXSP
LIS

RM3PUTERR
LIS

RM3PUTUPD
LIS

RM3SPLUDR
LIS

RM3RRV
LIS

RM3ROOT
LIS

RM3PUT
LIS