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

FILEID**RM3IUDR

```
RRRRRRRR    MM      MM    333333      IIIIII    UU      UU  DDDDDDDD    RRRRRRRR
RRRRRRRR    MM      MM    333333      IIIIII    UU      UU  DDDDDDDD    RRRRRRRR
RR      RR  MMMM  MMMM  33      33      II      UU      UU  DD      DD  RR      RR
RR      RR  MMMM  MMMM  33      33      II      UU      UU  DD      DD  RR      RR
RR      RR  MM  MM  MM          33      II      UU      UU  DD      DD  RR      RR
RR      RR  MM  MM  MM          33      II      UU      UU  DD      DD  RR      RR
RRRRRRRR    MM      MM        33        II      UU      UU  DD      DD  RRRRRRRR
RRRRRRRR    MM      MM        33        II      UU      UU  DD      DD  RRRRRRRR
RR  RR      MM      MM          33      II      UU      UU  DD      DD  RR  RR
RR  RR      MM      MM          33      II      UU      UU  DD      DD  RR  RR
RR    RR    MM      MM  33      33      II      UU      UU  DD      DD  RR    RR      ....
RR    RR    MM      MM  33      33      II      UU      UU  DD      DD  RR    RR      ....
RR      RR  MM      MM    333333      IIIIII    UUUUUUUUUU  DDDDDDDD    RR      RR    ....
RR      RR  MM      MM    333333      IIIIII    UUUUUUUUUU  DDDDDDDD    RR      RR    ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
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 RM3IUDR (LANGUAGE (BLISS32) ,
;     2          0002  0                  IDENT = 'V04-000'
;     3          0003  0                  ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !**************************************************************************
;     7          0007  1 !*                                                                        *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                  *
;    11          0011  1 !*                                                                        *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
;    17          0017  1 !*  TRANSFERRED.                                                           *
;    18          0018  1 !*                                                                        *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
;    21          0021  1 !*  CORPORATION.                                                           *
;    22          0022  1 !*                                                                        *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !*                                                                        *
;    27          0027  1 !**************************************************************************
;    28          0028  1
;    29          0029  1 !++
;    30          0030  1 !
;    31          0031  1 ! FACILITY:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !                INSERT USER DATA RECORD
;    35          0035  1 !
;    36          0036  1 !
;    37          0037  1 ! ENVIRONMENT:
;    38          0038  1 !
;    39          0039  1 !                VAX/VMS OPERATING SYSTEM
;    40          0040  1 !
;    41          0041  1 !--
;    42          0042  1 !
;    43          0043  1 !
;    44          0044  1 ! AUTHOR:        Wendy Koenig     CREATION DATE:                14-JUL-78  11:15
;    45          0045  1 !
;    46          0046  1 ! MODIFIED BY:
;    47          0047  1 !
;    48          0048  1 !       V03-012 JWT0174         Jim Teague                4-Apr-1984
;    49          0049  1 !               Fix one more key compression problem.  When a record
;    50          0050  1 !               to be inserted in a bucket won't fit, RMS first scans
;    51          0051  1 !               the bucket looking for deleted records whose space it
;    52          0052  1 !               can reclaim.  If a record is deleted, the position-of-
;    53          0053  1 !               insert of the new record is adjusted left the amount
;    54          0054  1 !               of the size of the deleted record.  Note however that
;    55          0055  1 !               the record following the record just deleted may have
;    56          0056  1 !               had it's key expanded as a result.  That amount is also
;    57          0057  1 !               taken into consideration when it comes to figuring the
```

```
;    58          0058  1 !                position-of-insert.
;    59          0059  1 !                    Keep in mind that this position-for-insert adjustment
;    60          0060  1 !                is only done for records before the position-for-insert.
;    61          0061  1 !                When deletion of a record results in position-for-insert
;    62          0062  1 !                being equal to where the deleted record used to be, no
;    63          0063  1 !                key expansion adjustment should be done.  This was happening
;    64          0064  1 !                in the case of a new record's position-of-insert being
;    65          0065  1 !                just after a deleted record, and as a result the position-
;    66          0066  1 !                of-insert became the middle of the record after the deleted
;    67          0067  1 !                record.
;    68          0068  1 !
;    69          0069  1 !        V03-011 MCN0016         Maria del C. Nasr       22-Mar-1983
;    70          0070  1 !                More linkages reorganization
;    71          0071  1 !
;    72          0072  1 !        V03-010 MCN0015         Maria del C. Nasr       24-Feb-1983
;    73          0073  1 !                Reorganize linkages
;    74          0074  1 !
;    75          0075  1 !        V03-009 TMK0005         Todd M. Katz            08-Jan-1983
;    76          0076  1 !                Add support for Recovery Unit Journalling and RU ROLLBACK
;    77          0077  1 !                Recovery of ISAM files.
;    78          0078  1 !
;    79          0079  1 !                This requires modification to the local routine RM$DEL_AND_TRY
;    80          0080  1 !                - the routine which scans a primary data bucket attempting to
;    81          0081  1 !                reclaim sufficient space so as to make room in the bucket for
;    82          0082  1 !                the insertion of a new record. This routine now has the ability
;    83          0083  1 !                to deal with records that have been modified (deleted or
;    84          0084  1 !                updated) within Recovery Units under a certain set of
;    85          0085  1 !                circumstances.
;    86          0086  1 !
;    87          0087  1 !                The global routine RM$INSERT_UDR must be modified so that if
;    88          0088  1 !                the primary data record must be repacked, the record size is
;    89          0089  1 !                increased by two after repacking if the state bit
;    90          0090  1 !                IRB$V_RU_UPDATE is set. This is necessary to allow for the
;    91          0091  1 !                primary data record to have two record size fields and be in a
;    92          0092  1 !                special format when it is eventually built.
;    93          0093  1 !
;    94          0094  1 !                The global routine RM$BLDUDR must also be modified so that
;    95          0095  1 !                records being built as the result of $UPDATEs are built in a
;    96          0096  1 !                special format when the IRB$V_RU_UPDATE state bit is set. This
;    97          0097  1 !                special format has two record size fields. The first size field
;    98          0098  1 !                is part of the record overhead and is the size of the amount of
;    99          0099  1 !                space the record reserves in case the Recovery Unit has to be
;   100          0100  1 !                aborted. The second size field occupies the last two bytes in
;   101          0101  1 !                the reserved space of the record and contains the actual size
;   102          0102  1 !                of the record.
;   103          0103  1 !
;   104          0104  1 !        V03-008 TMK0004         Todd M. Katz            06-Jan-1983
;   105          0105  1 !                Fixed a bug in the routine RM$DEL_AND_TRY. If this routine finds
;   106          0106  1 !                a record that it can delete (the record is marked deleted and
;   107          0107  1 !                duplicates are not allowed), then it reclaims the space it
;   108          0108  1 !                occupied by calling RM$DELETE_UDR. It then must adjust the
;   109          0109  1 !                address of the point of insertion of the new record provided
;   110          0110  1 !                the address of the reclaimed record preceeded the address of the
;   111          0111  1 !                record in the bucket. What this adjustment was not taking into
;   112          0112  1 !                account is that if primary key compression is enabled, the size
;   113          0113  1 !                of the key of the following record might change, affecting where
;   114          0114  1 !                the address of the point of insertion of the new record should
```

```
;  115     0115  1 !               be. This fix insures that such a change in key size is taken
;  116     0116  1 !               into account when the address of the point of insertion of the
;  117     0117  1 !               new record is adjusted.
;  118     0118  1 !
;  119     0119  1 !       V03-007 TMK0003         Todd M. Katz            14-Nov-1982
;  120     0120  1 !               Fixed a bug in the routine RM$DEL_AND_TRY. If this routine finds
;  121     0121  1 !               a record that it can delete (the record is marked deleted and
;  122     0122  1 !               duplicates are not allowed), then it reclaims the space it
;  123     0123  1 !               occupied by calling RM$DELETE_UDR. It then must adjust the
;  124     0124  1 !               address of the point of insertion of the new record provided
;  125     0125  1 !               the address of the reclaimed record preceeded the address of the
;  126     0126  1 !               record in the bucket. This was being done by adjusting the
;  127     0127  1 !               point of insertion by the difference in the bucket freespace
;  128     0128  1 !               offset pointer before and after the deleted record's space was
;  129     0129  1 !               reclaimed taking into account whether a RRV was created to
;  130     0130  1 !               replace it or not. This method is incorrect because it does not
;  131     0131  1 !               take into account the possibility that the key of the record
;  132     0132  1 !               following the deleted record might expand when primary key
;  133     0133  1 !               compression is enabled and the deleted record is removed. What
;  134     0134  1 !               is done now is to compute the amount of space occupied by the
;  135     0135  1 !               deleted record and just subtract that from the address of the
;  136     0136  1 !               point of insertion of the new record when necessary.
;  137     0137  1 !
;  138     0138  1 !       V03-006 KBT0167         Keith B. Thompson       23-Aug-1982
;  139     0139  1 !               Reorganize psects
;  140     0140  1 !
;  141     0141  1 !       V03-005 TMK0002         Todd M. Katz            08-Aug-1982
;  142     0142  1 !               Re-write the routine DEL_AND_TRY. The $DELETE operation has been
;  143     0143  1 !               completely re-written and the interfacing of this routine to
;  144     0144  1 !               the routines involved has drastically changed.
;  145     0145  1 !
;  146     0146  1 !       V03-004 TMK0001         Todd M. Katz            02-Jul-1982
;  147     0147  1 !               Implement the RMS cluster solution for next record positioning.
;  148     0148  1 !               As the next record positioning context is now kept locally
;  149     0149  1 !               within the IRAB, it is no loner necessary to reference the
;  150     0150  1 !               NRP cell, a structure whose existance has been terminated, in
;  151     0151  1 !               order to both set and retrieve the RFA address of the user
;  152     0152  1 !               data record being inserted. Always reference the RFA of the new
;  153     0153  1 !               (updated) record by means of the subfields IRB$L_PUTUP_VBN
;  154     0154  1 !               and IRB$W_PUTUP_ID.
;  155     0155  1 !
;  156     0156  1 !       V03-003 KBT0073         Keith B. Thompson       28-Jun-1982
;  157     0157  1 !               Modify del_and_try for the new NPR delete requirements
;  158     0158  1 !
;  159     0159  1 !       V03-002 MCN0014         Maria del C. Nasr       11-Jun-1982
;  160     0160  1 !               Eliminate overhead at end of data bucket that was to be
;  161     0161  1 !               used for duplicate continuation bucket processing.
;  162     0162  1 !
;  163     0163  1 !       V03-001 TMK0001         Todd M. Katz            14-March-1982
;  164     0164  1 !               Change the use of RM$INSERT_UDR's lone parameter so that
;  165     0165  1 !               it is both an input and an output parameter. This is
;  166     0166  1 !               because in one special case the size of the record to be
;  167     0167  1 !               inserted may change, but the insertion does not take place
;  168     0168  1 !               under the control of this routine. If there is insufficient
;  169     0169  1 !               room in the bucket for the record, an attempt is made to
;  170     0170  1 !               squish out the keys of all deleted records with keys currently
;  171     0171  1 !               in the bucket. If this is a prologue 3 file with compressed
```

```
;  172     0172  1 !               primary keys, and the record to be inserted follows such a
;  173     0173  1 !               deleted record, this means the record must also be repacked as
;  174     0174  1 !               its size could have changed. If there is still insufficient
;  175     0175  1 !               room in the bucket for the new record, this new size value
;  176     0176  1 !               must be returned, since a bucket split is to occur, and the
;  177     0177  1 !               insertion of the new record will take place elsewhere.
;  178     0178  1 !
;  179     0179  1 !       V02-016 DJD0001         Darrell Duffy           1-March-1982
;  180     0180  1 !               Fix reference to record buffer to prevent protection
;  181     0181  1 !               hole.
;  182     0182  1 !
;  183     0183  1 !       V02-015 PSK0001         Paulina S. Knibbe       08-Oct-1981
;  184     0184  1 !               Fix 014. When scanning a bucket for deleted records to
;  185     0185  1 !               squish, this routine was getting confused after
;  186     0186  1 !               successfully squishing a record which also caused
;  187     0187  1 !               the following key to be expanded (because of front-end
;  188     0188  1 !               compression).
;  189     0189  1 !
;  190     0190  1 !       V02-014 MCN0013         Maria del C. Nasr       04-Aug-1981
;  191     0191  1 !               When we delete records, and expand keys the position of insert
;  192     0192  1 !               must be updated to reflect characters moved.
;  193     0193  1 !
;  194     0194  1 !       V02-013 MCN0012         Maria del C. Nasr       07-Jul-1981
;  195     0195  1 !               Fix problem in which if a record was to be added after a record
;  196     0196  1 !               that was deleted by DEL_AND_TRY, the key compression did not
;  197     0197  1 !               match anymore.  Record must be packed again.
;  198     0198  1 !
;  199     0199  1 !       V02-012 MCN0010         Maria del C. Nasr       15-May-1981
;  200     0200  1 !               Make changes to be able to $PUT prologue 3 records.
;  201     0201  1 !
;  202     0202  1 !       V02-011 MCN0006         Maria del C. Nasr       13-Mar-1981
;  203     0203  1 !               Increase size of record identifier to a word in NRP.
;  204     0204  1 !
;  205     0205  1 !       V02-010 REFORMAT        Paulina S. Knibbe       23-JUL-80
;  206     0206  1 !
;  207     0207  1 ! REVISION HISTORY:
;  208     0208  1 !
;  209     0209  1 !   Wendy Koenig,       28-SEP-78  8:51
;  210     0210  1 !   X0002 - WHEN SQUISHING OUT DELETED RECORDS ALWAYS LEAVE A 2-BYTE RRV
;  211     0211  1 !
;  212     0212  1 !   Christiar Saether,  4-OCT-78  9:45
;  213     0213  1 !   X0003 - m ifications for UPDATE
;  214     0214  1 !
;  215     0215  1 !   Wendy Koenig,       12-OCT-78  15:56
;  216     0216  1 !   X0004 - IF ITS AN EMPTY BUCKET, FORCE RECORD ALWAYS TO FIT, REGARDLESS Of
;  217     0217  1 !   LOA BIT
;  218     0218  1 !
;  219     0219  1 !   Wendy Koenig,       24-OCT-78  14:02
;  220     0220  1 !   X0005 - MAKE CHANGES CAUSED BY SHARING CONVENTIONS
;  221     0221  1 !
;  222     0222  1 !   Christian Saether,  13-DEC-78  20:23
;  223     0223  1 !   X0006 - DEL_AND_TRY forces DELETE_UDR to always remove record
;  224     0224  1 !
;  225     0225  1 !   Wendy Koenig,       22-JAN-79  17:01
;  226     0226  1 !   X0007 - IGNORE LOA BIT IF UPDATE
;  227     0227  1 !
;  228     0228  1 !   Wendy Koenig,       25-JAN-79  11:25
```

```
;  229     0229  1 !   X0008 - GET RID OF SETTING VALID
;  230     0230  1 !
;  231     0231  1 !   Christian Saether, 1-Jan-80  21:55
;  232     0232  1 !   0009 - check for id available moved to rm$put3b from rm$insert_udr
;  233     0233  1 !   because it's not relevant in update situation (fixes bug splitting
;  234     0234  1 !   bucket on update when all id's are used)
;  235     0235  1 !
;  236     0236  1 !*****
;  237     0237  1
;  238     0238  1 LIBRARY 'RMSLIB:RMS';
;  239     0239  1
;  240     0240  1 REQUIRE 'RMSSRC:RMSIDXDEF';
;  241     0305  1
;  242     0306  1 ! Define default PSECTS for code
;  243     0307  1 !
;  244     0308  1 PSECT
;  245     0309  1     CODE = RM$RMS3(PSECT_ATTR),
;  246     0310  1     PLIT = RM$RMS3(PSECT_ATTR);
;  247     0311  1
;  248     0312  1 ! Linkages
;  249     0313  1 !
;  250     0314  1 LINKAGE
;  251     0315  1     L_JSB01,
;  252     0316  1     L_PRESERVE1,
;  253     0317  1     L_RABREG_567,
;  254     0318  1     L_RABREG_4567,
;  255     0319  1     L_RABREG_67,
;  256     0320  1     L_REC_OVHD,
;  257     0321  1
;  258     0322  1     ! Local linkages
;  259     0323  1     !
;  260     0324  1     RL$DEL_AND_TRY = JSB()
;  261     0325  1                    : GLOBAL(COMMON_IOREG,COMMON_RABREG,R_REC_ADDR,R_IDX_DFN);
;  262     0326  1
;  263     0327  1 ! Forward Routines
;  264     0328  1 !
;  265     0329  1 FORWARD ROUTINE
;  266     0330  1     RM$INSERT_REC       : RL$RABREG_456,
;  267     0331  1     RM$INSERT_UDR       : RL$RABREG_4567;
;  268     0332  1
;  269     0333  1 ! External Routines
;  270     0334  1 !
;  271     0335  1 EXTERNAL ROUTINE
;  272     0336  1     RM$DELETE_UDR       : RL$RABREG_4567,
;  273     0337  1     RM$GETNEXT_REC      : RL$RABREG_67,
;  274     0338  1     RM$MOVE             : RL$PRESERVE1,
;  275     0339  1     RM$PACK_REC         : RL$RABREG_567,
;  276     0340  1     RM$RECOMPR_KEY      : RL$JSB01,
;  277     0341  1     RM$REC_OVHD         : RL$REC_OVHD,                                  ; R
;  278     0342  1     RM$RU_RECLAIM       : RL$RABREG_67;
```

```
;  280          0343  1 %SBTTL  'RM$BLDUDR'
;  281          0344  1 GLOBAL ROUTINE RM$BLDUDR (RECSZ) : RL$RABREG_4567 =
;  282          0345  1
;  283          0346  1 !++
;  284          0347  1 !
;  285          0348  1 ! FUNCTIONAL DESCRIPTION:
;  286          0349  1 !
;  287          0350  1 ! insert the user's data record into the bucket w/ all its overhead
;  288          0351  1 !
;  289          0352  1 ! CALLING SEQUENCE:
;  290          0353  1 !
;  291          0354  1 !       BSBW RM$BLDUDR()
;  292          0355  1 !
;  293          0356  1 ! INPUT PARAMETERS:
;  294          0357  1 !       RECSZ - record size of record to be inserted including overhead
;  295          0358  1 !
;  296          0359  1 ! IMPLICIT INPUTS:
;  297          0360  1 !       REC_ADDR -- pointer to place to insert record
;  298          0361  1 !       BKT_ADDR -- nxtrecid field
;  299          0362  1 !       IDX_DFN -- index descriptor for data bucket type
;  300          0363  1 !       BDB -- vbn of bucket
;  301          0364  1 !       RAB -- rsz, rbf fields
;  302          0365  1 !       IFAB -- rfm field,
;  303          0366  1 !       IRAB -- mode field, V_RU_UPDATE
;  304          0367  1 !
;  305          0368  1 ! OUTPUT PARAMETERS:
;  306          0369  1 !       NONE
;  307          0370  1 !
;  308          0371  1 ! IMPLICIT OUTPUTS:
;  309          0372  1 !       record is inserted into bucket, nxtrecid is incremented if new record
;  310          0373  1 !       REC_ADDR points to the first byte of the next record
;  311          0374  1 !       IRB$L_PUTUP_VBN, and IRB$W_PUTUP_ID are filled in with the RFA address
;  312          0375  1 !           of the record
;  313          0376  1 !       IRB$V_RU_UPDATE is always cleared.
;  314          0377  1 !
;  315          0378  1 ! ROUTINE VALUE:
;  316          0379  1 !       RMSSUC
;  317          0380  1 !
;  318          0381  1 ! SIDE EFFECTS:
;  319          0382  1 !
;  320          0383  1 !       Record is inserted into bucket.
;  321          0384  1 !       If the state bit IRB$V_RU_UPDATE is set, the record is built in a
;  322          0385  1 !           special format in that it will contain two record size fields. The
;  323          0386  1 !           amount of space the record occupies will be found in the record
;  324          0387  1 !           overhead's size field while the true size of the record will be
;  325          0388  1 !           found in the last two bytes of this reserved space.
;  326          0389  1 !
;  327          0390  1 !--
;  328          0391  1
;  329          0392  2     BEGIN
;  330          0393  2
;  331          0394  2     BUILTIN
;  332          0395  2         TESTBITSC;
;  333          0396  2
;  334          0397  2     EXTERNAL REGISTER
;  335          0398  2         COMMON_IO_STR,
;  336          0399  2         R_REC_ADDR_STR,
```

```
;  337     0400  2          R_IDX_DFN_STR,
;  338     0401  2          R_IFAB_STR,
;  339     0402  2          R_IRAB_STR,
;  340     0403  2          R_RAB_STR;
;  341     0404  2
;  342     0405  2      IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  343     0406  2      THEN
;  344     0407  3          BEGIN
;  345     0408  3
;  346     0409  3          IF NOT .IRAB[IRB$V_UPDATE]
;  347     0410  3          THEN
;  348     0411  3
;  349     0412  3              ! this is a put operation so the VBN and ID fields for this record must
;  350     0413  3              ! be filled in the record pointer fields to build the record
;  351     0414  3              !
;  352     0415  4              BEGIN
;  353     0416  4
;  354     0417  4              IF .BDB NEQ .IRAB[IRB$L_CURBDB]
;  355     0418  4
;  356     0419  4                  ! the record is going into a new bucket so zero the ID to
;  357     0420  4                  ! signal this. the ID's will get reassigned further on anyway
;  358     0421  4                  !
;  359     0422  4              THEN
;  360     0423  4                  IRAB[IRB$W_LAST_ID] = 0
;  361     0424  4              ELSE
;  362     0425  4
;  363     0426  4                  ! the record goes into the original bucket so use the next ID
;  364     0427  4                  !
;  365     0428  5                  BEGIN
;  366     0429  5                  IRAB[IRB$W_LAST_ID] = .BKT_ADDR[BKT$B_NXTRECID];
;  367     0430  5                  IRAB[IRB$W_PUTUP_ID] = .BKT_ADDR[BKT$B_NXTRECID];
;  368     0431  5                  BKT_ADDR[BKT$B_NXTRECID] = .BKT_ADDR[BKT$B_NXTRECID] + 1;
;  369     0432  4                  END;
;  370     0433  4
;  371     0434  4              IRAB[IRB$L_PUTUP_VBN] = .BDB[BDB$L_VBN];
;  372     0435  3              END;
;  373     0436  3
;  374     0437  3          REC_ADDR[IRC$B_CONTROL] = 2;
;  375     0438  3
;  376     0439  3          ! fill in record ID and back pointer ID fields, being sure to use
;  377     0440  3          ! the original ID if this is an update case
;  378     0441  3          !
;  379     0442  3          REC_ADDR[IRC$B_ID] = .IRAB[IRB$W_LAST_ID];
;  380     0443  3          REC_ADDR[IRC$B_RRV_ID] = .IRAB[IRB$W_PUTUP_ID];
;  381     0444  3          REC_ADDR = .REC_ADDR + 3;
;  382     0445  3
;  383     0446  3          (.REC_ADDR) = .IRAB[IRB$L_PUTUP_VBN];
;  384     0447  3          REC_ADDR = .REC_ADDR + 4;
;  385     0448  3
;  386     0449  3          ! if not fixed length records, move size field in
;  387     0450  3          !
;  388     0451  3
;  389     0452  3          IF .IFAB[IFB$B_RFMORG] NEQ FAB$C_FIX
;  390     0453  3          THEN
;  391     0454  4              BEGIN
;  392     0455  4              (.REC_ADDR)<0, 16> = .RAB[RAB$W_RSZ];
;  393     0456  4              REC_ADDR = .REC_ADDR + IRC$C_DATSZFLD;
```

```
;  394     0457  3              END;
;  395     0458  3
;  396     0459  3          ! move user's data record in                                                    ;
;  397     0460  3          !
;  398     0461  4          BEGIN
;  399     0462  4
;  400     0463  4          GLOBAL REGISTER
;  401     0464  4              R_IMPURE;
;  402     0465  4
;  403     0466  4          REC_ADDR = RM$MOVE (.IRAB[IRB$W_RSZ], .IRAB[IRB$L_RBF], .REC_ADDR);
;  404     0467  3          END;
;  405     0468  3          END
;  406     0469  3
;  407     0470  2      ELSE
;  408     0471  3          BEGIN
;  409     0472  3
;  410     0473  3          IF NOT .IRAB[IRB$V_UPDATE]
;  411     0474  3          THEN
;  412     0475  3
;  413     0476  3              ! this is a put operation so the VBN and ID fields for this record must
;  414     0477  3              ! be filled in the record pointer fields to build the record
;  415     0478  3              !
;  416     0479  4              BEGIN
;  417     0480  4
;  418     0481  4              IF .BDB NEQ .IRAB[IRB$L_CURBDB]
;  419     0482  4
;  420     0483  4                  ! the record is going into a new bucket so zero the ID to signal
;  421     0484  4                  ! this. the ID's will get reassigned further on anyway
;  422     0485  4                  !
;  423     0486  4              THEN
;  424     0487  4                  IRAB[IRB$W_LAST_ID] = 0
;  425     0488  4              ELSE
;  426     0489  4
;  427     0490  4                  ! the record goes into the original bucket so use the next ID
;  428     0491  4                  !
;  429     0492  5                  BEGIN
;  430     0493  5                  IRAB[IRB$W_LAST_ID] = .BKT_ADDR[BKT$W_NXTRECID];
;  431     0494  5                  IRAB[IRB$W_PUTUP_ID] = .BKT_ADDR[BKT$W_NXTRECID];
;  432     0495  5                  BKT_ADDR[BKT$W_NXTRECID] = .BKT_ADDR[BKT$W_NXTRECID] + 1;
;  433     0496  4                  END;
;  434     0497  4
;  435     0498  4              IRAB[IRB$L_PUTUP_VBN] = .BDB[BDB$L_VBN];
;  436     0499  3              END;
;  437     0500  3
;  438     0501  3          ! Fill in the pointer size field
;  439     0502  3          !
;  440     0503  3          REC_ADDR[IRC$B_CONTROL] = 2;
;  441     0504  3
;  442     0505  3          ! If this record is to be in a special format then set the appropriate
;  443     0506  3          ! record control bit.
;  444     0507  3          !
;  445     0508  3          IF .IRAB[IRB$V_RU_UPDATE]
;  446     0509  3          THEN
;  447     0510  3              REC_ADDR[IRC$V_RU_UPDATE] = 1;
;  448     0511  3
;  449     0512  3          ! fill in record ID and back pointer ID fields, being sure to use
;  450     0513  3          ! the original ID if this is an update case.  Also, move VBN into
```

```
;  451     0514  3          ! record.
;  452     0515  3          !
;  453     0516  3          REC_ADDR[IRC$W_ID] = .IRAB[IRB$W_LAST_ID];
;  454     0517  3          REC_ADDR[IRC$W_RRV_ID] = .IRAB[IRB$W_PUTUP_ID];
;  455     0518  3          REC_ADDR = .REC_ADDR + 5;
;  456     0519  3          (.REC_ADDR) = .IRAB[IRB$L_PUTUP_VBN];
;  457     0520  3          REC_ADDR = .REC_ADDR + 4;
;  458     0521  3          RECSZ = .RECSZ - IRC$C_FIXOVHSZ3;
;  459     0522  3
;  460     0523  3          ! If not fixed length records, or fixed length compressed records
;  461     0524  3          ! move size field in
;  462     0525  3          !
;  463     0526  3          IF .IFAB[IFB$B_RFMORG] NEQ FAB$C_FIX
;  464     0527  4            OR (.IFAB[IFB$B_RFMORG] EQL FAB$C_FIX
;  465     0528  4                AND .IDX_DFN[IDX$B_DATBKTYP] NEQU IDX$C_NCMPNCMP)
;  466     0529  3          THEN
;  467     0530  4              BEGIN
;  468     0531  4              RECSZ = .RECSZ - IRC$C_DATSZFLD;
;  469     0532  4              (.REC_ADDR)<0, 16> = .RECSZ;
;  470     0533  4              REC_ADDR = .REC_ADDR + IRC$C_DATSZFLD;
;  471     0534  4
;  472     0535  4              ! If the record is to be in the special format, then reduce record
;  473     0536  4              ! size by the two bytes that were added to it to allow for the
;  474     0537  4              ! second record size field, and move the true size of the record
;  475     0538  4              ! into this second record size field (which occupies the last two
;  476     0539  4              ! bytes in the reserved space of the record).
;  477     0540  4              !
;  478     0541  4              IF .IRAB[IRB$V_RU_UPDATE]
;  479     0542  4              THEN
;  480     0543  5                  BEGIN
;  481     0544  5                  RECSZ = .RECSZ - IRC$C_DATSZFLD;
;  482     0545  5                  (.REC_ADDR + .RECSZ)<0,16> = .RECSZ;
;  483     0546  4                  END;
;  484     0547  3              END;
;  485     0548  3
;  486     0549  3              ! Move user's data record in.
;  487     0550  3              !
;  488     0551  4              BEGIN
;  489     0552  4
;  490     0553  4              GLOBAL REGISTER
;  491     0554  4                  R_IMPURE;
;  492     0555  4
;  493     0556  4              REC_ADDR = RM$MOVE(.RECSZ, .IRAB[IRB$L_RECBUF], .REC_ADDR);
;  494     0557  3              END;
;  495     0558  3
;  496     0559  3              ! If the record is in a special format, then increment REC_ADDR by the
;  497     0560  3              ! size of the additional record size field so that it will point to the
;  498     0561  3              ! end of the special data record.
;  499     0562  3              !
;  500     0563  3              IF TESTBITSC (IRAB[IRB$V_RU_UPDATE])
;  501     0564  3              THEN
;  502     0565  3                  REC_ADDR = .REC_ADDR + IRC$C_DATSZFLD;
;  503     0566  2              END;
;  504     0567  2
;  505     0568  3          RETURN RMSSUC()
;  506     0569  3
;  507     0570  1          END;                                    ! { end of routine }
```

```
                                        .TITLE  RM3IUDR
                                        .IDENT  \V04-000\

                                        .EXTRN  RM$DELETE_UDR, RM$GETNEXT_REC
                                        .EXTRN  RM$MOVE, RM$PACK_REC
                                        .EXTRN  RM$RECOMPR_KEY, RM$REC_OVHD
                                        .EXTRN  RM$RU_RECLAIM

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

                                5B  DD 00000 RM$BLDUDR::
                                            PUSHL   R11                             ; 0344
                          03    00B7  CA  91 00002        CMPB    183(IFAB), #3       ; 0405
                                      52  1E 00007        BGEQU   5$
                    1E    06    A9        03  E0 00009   BBS     #3, 6(IRAB), 3$     ; 0409
                          20    A9        54  D1 0000E   CMPL    BDB, 32(IRAB)       ; 0417
                                      05  13 00012        BEQL    1$
                                74    A9  B4 00014        CLRW    116(IRAB)           ; 0423
                                      0E  11 00017        BRB     2$
                          74    A9        06  A5 9B 00019 1$:    MOVZBW  6(BKT_ADDR), 116(IRAB)  ; 0429
                        0080    C9        06  A5 9B 0001E       MOVZBW  6(BKT_ADDR), 128(IRAB)  ; 0430
                                06  A5  96 00024        INCB    6(BKT_ADDR)         ; 0431
                          78    A9  1C    A4  D0 00027 2$:    MOVL    28(BDB), 120(IRAB)  ; 0434
                                86        02  90 0002C 3$:    MOVB    #2, (REC_ADDR)+     ; 0437
                                86  74    A9  90 0002F        MOVB    116(IRAB), (REC_ADDR)+  ; 0442
                                86  0080  C9  90 00033        MOVB    128(IRAB), (REC_ADDR)+  ; 0443
                                86  78    A9  D0 00038        MOVL    120(IRAB), (REC_ADDR)+  ; 0446
                                01  50    AA  91 0003C        CMPB    80(IFAB), #1        ; 0452
                                            04  13 00040        BEQL    4$
                                86  22    A8  B0 00042        MOVW    34(RAB), (REC_ADDR)+    ; 0455
                                56  DD 00046 4$:     PUSHL   REC_ADDR                ; 0466
                                58  A9  DD 00048        PUSHL   88(IRAB)
                                7E  56  A9  3C 0004B        MOVZWL  86(IRAB), -(SP)
                                      0000G 30  0004F       BSBW    RM$MOVE
                                5E        0C  C0 00052        ADDL2   #12, SP
                                56        50  D0 00055        MOVL    R0, REC_ADDR
                                        0084  31 00058        BRW     12$             ; 0405
                    1   06    A9        03  E0 0005B 5$:     BBS     #3, 6(IRAB), 8$     ; 0473
                          20    A9        54  D1 00060        CMPL    BDB, 32(IRAB)       ; 0481
                                      05  13 00064        BEQL    6$
                                74    A9  B4 00066        CLRW    116(IRAB)           ; 0487
                                      0E  11 00069        BRB     7$
                          74    A9        06  A5 B0 0006B 6$:    MOVW    6(BKT_ADDR), 116(IRAB)  ; 0493
                        0080    C9        06  A5 B0 00070       MOVW    6(BKT_ADDR), 128(IRAB)  ; 0494
                                06  A5  B6 00076        INCW    6(BKT_ADDR)         ; 0495
                          78    A9  1C    A4  D0 00079 7$:    MOVL    28(BDB), 120(IRAB)  ; 0498
                                06        02  90 0007E 8$:    MOVB    #2, (REC_ADDR)      ; 0503
                                07  A9  95 00081        TSTB    7(IRAB)             ; 0508
                                            04  18 00084        BGEQ    9$
                                66  40    8F  88 00086        BISB2   #64, (REC_ADDR)     ; 0510
                          01    A6  74    A9  B0 0008A 9$:    MOVW    116(IRAB), 1(REC_ADDR)  ; 0516
                          03    A6  0080  C9  B0 0008F        MOVW    128(IRAB), 3(REC_ADDR)  ; 0517
                                56        05  C0 00095        ADDL2   #5, REC_ADDR        ; 0518
                                86  78    A9  D0 00098        MOVL    120(IRAB), (REC_ADDR)+  ; 0519
                          08    AE        09  C2 0009C        SUBL2   #9, RECSZ           ; 0521
```

```
                                 01        50    AA  91 000A0          CMPB    80(IFAB), #1                                    ; 0526
                                           06    12  00A4             BNEQ    10$                                             ;
                                 06        29    A7  91 000A6          CMPB    41(IDX_DFN), #6                                 ; 0528
                                           1A    13  000AA             BEQL    11$                                             ;
                              08 AE              02  C2 000AC 10$:     SUBL2   #2, RECSZ                                       ; 0531
                                 86        09    AE  B0 000B0          MOVW    RECSZ, (REC_ADDR)+                              ; 0532
                                           07    A9  95 000B4          TSTB    7(IRAB)                                         ; 0541
                                           0D    18  000B7             BGEQ    11$                                             ;
                              08 AE              02  C2 000B9          SUBL2   #2, RECSZ                                       ; 0544
                        50                 56    08  AE  C1 000BD     ADDL3   RECSZ, REC_ADDR, R0                             ; 0545
                                           60    08  AE  B0 000C2     MOVW    RECSZ, (R0)                                     ;
                                                 56  DD 000C6 11$:     PUSHL   REC_ADDR                                        ; 0556
                                           68    A9  DD 000C8          PUSHL   104(IRAB)                                       ;
                                           10    AE  DD 000CB          PUSHL   RECSZ                                           ;
                                              0000G    30 000CE        BSBW    RM$MOVE                                         ;
                                 5E                0C  C0 000D1       ADDL2   #12, SP                                         ;
                                 56                50  D0 000D4       MOVL    R0, REC_ADDR                                    ;
                        03       04        A9          1F  E5 000D7    BBCC    #31, 4(IRAB), 12$                               ; 0563
                                 56                02  C0 000DC       ADDL2   #2, REC_ADDR                                    ; 0565
                                 50                01  D0 000DF 12$:   MOVL    #1, R0                                          ; 0568
                                                0800 8F  BA 000E2      POPR    #^M<R11>                                        ; 0570
                                                         05 000E6      RSB                                                     ;

; Routine Size:  231 bytes,    Routine Base:  RM$RMS3 + 0000
```

```
;  509     0571  1 %SBTTL  'RMSDEL_AND_TRY'
;  510     0572  1 ROUTINE RMSDEL_AND_TRY : RL$DEL_AND_TRY =
;  511     0573  1
;  512     0574  1 !++
;  513     0575  1 !
;  514     0576  1 ! FUNCTIONAL DESCRIPTION:
;  515     0577  1 !
;  516     0578  1 !       If duplicate primary keys are not allowed, this routine scans the
;  517     0579  1 !       current primary data bucket for primary data records that are just
;  518     0580  1 !       marked deleted, and deletes any that it encounters. If records are
;  519     0581  1 !       encountered during the bucket scan which were modified within a
;  520     0582  1 !       Recovery Unit, then they maybe subjected to special processing provided
;  521     0583  1 !       the Recovery Unit in which they were modified has completed. Records
;  522     0584  1 !       that were deleted within a Recovery Unit may have their space reclaimed,
;  523     0585  1 !       and records that were updated may be reformated.
;  524     0586  1 !
;  525     0587  1 !       If duplicate primary keys are allowed this routine can not reclaim the
;  526     0588  1 !       space occupued by records that are just marked deleted because of
;  527     0589  1 !       constraints imposed by the RMS cluster solution for next record
;  528     0590  1 !       positioning. However, if the file is RU Journallable, then the bucket
;  529     0591  1 !       scan is done anyway so that any records modified within recovery units
;  530     0592  1 !       can be processed appropriately.
;  531     0593  1 !
;  532     0594  1 !       Whenever a deleted record is encountered, is is completely removed, a
;  533     0595  1 !       two-byte deleted RRV without pointer is created for it at the end of the
;  534     0596  1 !       bucket if the file is not a prologue 3 file and the record is in its
;  535     0597  1 !       original bucket, and the bucket's freespace is appropriately updated.
;  536     0598  1 !       Because this routine is only called whenever there is insufficient room
;  537     0599  1 !       in the primary data bucket for the insertion of a new record, the
;  538     0600  1 !       point of insertion of the new record must also be updated whenever a
;  539     0601  1 !       deleted record is eliminated, and the position of the deleted record
;  540     0602  1 !       had preceeded the point of insertion of the new record in the bucket.
;  541     0603  1 !
;  542     0604  1 !       If the file is Recovery Unit Journallable, then the RRV records at the
;  543     0605  1 !       end of the bucket will also be scanned looking for those records that
;  544     0606  1 !       were deleting within a completed Recovery Unit. If such records are
;  545     0607  1 !       found they are deleted for good at this time.
;  546     0608  1 !
;  547     0609  1 !
;  548     0610  1 ! CALLING SEQUENCE:
;  549     0611  1 !
;  550     0612  1 !       RMSDEL_AND_TRY()
;  551     0613  1 !
;  552     0614  1 ! INPUT PARAMETERS:
;  553     0615  1 !       NONE
;  554     0616  1 !
;  555     0617  1 ! IMPLICIT INPUTS:
;  556     0618  1 !
;  557     0619  1 !       BKT_ADDR                - address of primary data bucket
;  558     0620  1 !           BKT$W_FREESPACE     - offset pointer to freespace in bucket
;  559     0621  1 !
;  560     0622  1 !       IDX_DFN                 - index descriptor for primary key of reference
;  561     0623  1 !           IDX$V_DUPKEYS       - if set, duplicate keys are allowed
;  562     0624  1 !           IDX$V_KEY_COMPR     - if set, primary key compression is enabled
;  563     0625  1 !
;  564     0626  1 !       IFAB                    - address of IFAB
;  565     0627  1 !           IFB$B_PLG_VER       - prologue version of file
```

```
;  566     0628  1 !           IFB$V_RU            - if set, file is RU Journallable
;  567     0629  1 !
;  568     0630  1 !       REC_ADDR                - address of point of insertion of new record
;  569     0631  1 !
;  570     0632  1 ! OUTPUT PARAMETERS:
;  571     0633  1 !       NONE
;  572     0634  1 !
;  573     0635  1 ! IMPLICIT OUTPUTS:
;  574     0636  1 !
;  575     0637  1 !       IRAB                    - address of IRAB
;  576     0638  1 !           IRB$W_POS_INS       - offset to point of insertion of new record
;  577     0639  1 !
;  578     0640  1 !       REC_ADDR                - address of point of insertion of new record
;  579     0641  1 !
;  580     0642  1 ! ROUTINE VALUE:
;  581     0643  1 !
;  582     0644  1 !       0 if no records were deleted
;  583     0645  1 !       1 if some records were deleted
;  584     0646  1 !
;  585     0647  1 ! SIDE EFFECTS:
;  586     0648  1 !
;  587     0649  1 !       AP is trashed.
;  588     0650  1 !       If duplicate primary keys are not allowed, and deleted records were
;  589     0651  1 !           found in the bucket they were completely deleted, and the bucket
;  590     0652  1 !           freespace offset and position of insertion of the new record
;  591     0653  1 !           updated appropriately.
;  592     0654  1 !       If this is a prologue 2 file then any deleted records encountered that
;  593     0655  1 !           were in their original bucket have a deleted RRV (without a RRV
;  594     0656  1 !           pointer) created for it at the end of the bucket to reserve the ID
;  595     0657  1 !           so it can not be recycled.
;  596     0658  1 !       Any records that had been deleted within Recovery Units might have been
;  597     0659  1 !           deleted for good and had their space reclaimed.
;  598     0660  1 !       Any records that had been updated within Recovery Units might have been
;  599     0661  1 !           reformated.
;  600     0662  1 !
;  601     0663  1 !--
;  602     0664  1
;  603     0665  2     BEGIN
;  604     0666  2
;  605     0667  2     BUILTIN
;  606     0668  2         AP,
;  607     0669  2         TESTBITSC;
;  608     0670  2
;  609     0671  2     EXTERNAL REGISTER
;  610     0672  2         COMMON_IO_STR,
;  611     0673  2         COMMON_RAB_STR,
;  612     0674  2         R_IDX_DFN_STR,
;  613     0675  2         R_REC_ADDR_STR;
;  614     0676  2
;  615     0677  2     LOCAL
;  616     0678  2         FLAGS           : BLOCK [1],
;  617     0679  2         POS_INSERT;
;  618     0680  2
;  619     0681  2     MACRO
;  620     0682  2         KEY_EXPANSION   = 0,0,1,0 %,
;  621     0683  2         SPACE_RECLAIMED = 0,1,1,0 %;
;  622     0684  2
```

```
;  623     0685  2      ! If the file allows duplicate primary keys then the space occupied by
;  624     0686  2      ! deleted records can not be recover on-line due to constraints imposed
;  625     0687  2      ! by the RMS cluster solution to next record positioning. Avoid the
;  626     0688  2      ! overhead of the bucket scan, unless the file is RU Journallable in which
;  627     0689  2      ! case perform the bucket scan so as to process those records which had
;  628     0690  2      ! been deleted within recovery units.
;  629     0691  2      !
;  630     0692  2      IF  .IDX_DFN[IDX$V_DUPKEYS]
;  631     0693  2          AND
;  632     0694  2          NOT .IFAB[IFB$V_RU]
;  633     0695  2      THEN
;  634     0696  2          RETURN 0
;  635     0697  2      ELSE
;  636     0698  2          FLAGS = 0;
;  637     0699  2
;  638     0700  2      ! Prepare to scan the bucket for deleted records by saving the address of
;  639     0701  2      ! the point of insertion of the new record and initializing REC_ADDR to the
;  640     0702  2      ! address of the very first record in the primary data bucket.
;  641     0703  2      !
;  642     0704  2      POS_INSERT    = .REC_ADDR;
;  643     0705  2      REC_ADDR      = .BKT_ADDR + BKT$C_OVERHDSZ;
;  644     0706  2
;  645     0707  2      ! Scan the entire primary data bucket searching for primary data records
;  646     0708  2      ! that are just marked deleted. The search will terminate either when all
;  647     0709  2      ! records in the bucket have been exhausted, or the first RRV in the bucket
;  648     0710  2      ! is encountered (NOTE, if the file is Recovery Unit Journallable, then the
;  649     0711  2      ! scan will terminate only when every record in the bucket has been looked
;  650     0712  2      ! at including the RRVs).
;  651     0713  2      !
;  652     0714  4      WHILE ((.REC_ADDR LSSA (.BKT_ADDR + .BKT_ADDR[BKT$W_FREESPACE]))
;  653     0715  3              AND
;  654     0716  4             (NOT .REC_ADDR[IRC$V_RRV]
;  655     0717  4                   OR
;  656     0718  3                   .IFAB[IFB$V_RU]))
;  657     0719  2      DO
;  658     0720  3          BEGIN
;  659     0721  3
;  660     0722  3          ! If the current record has been modified within a Recovery Unit then it
;  661     0723  3          ! may require special processing depending upon how the record was
;  662     0724  3          ! modified and whether the Recovery Unit terminated successfully or is
;  663     0725  3          ! still in progress.
;  664     0726  3          !
;  665     0727  3          IF .REC_ADDR[IRC$V_RU_UPDATE]
;  666     0728  3              OR
;  667     0729  3              .REC_ADDR[IRC$V_RU_DELETE]
;  668     0730  3          THEN
;  669     0731  4              BEGIN
;  670     0732  4
;  671     0733  4              LOCAL
;  672     0734  4                  OLD_FREESPACE : WORD;
;  673     0735  4
;  674     0736  4              ! Save the current freespace offset pointer into the primary data
;  675     0737  4              ! bucket.
;  676     0738  4              !
;  677     0739  4              OLD_FREESPACE = .BKT_ADDR[BKT$W_FREESPACE];
;  678     0740  4
;  679     0741  4              ! If it was possible to reclaim any space at all from the RU
```

```
;  680     0742  4              ! modified record, then set the appropriate state bit and adjust
;  681     0743  4              ! the position of insertion of the new record if necessary.
;  682     0744  4              !
;  683     0745  4              IF RM$RU_RECLAIM()
;  684     0746  4              THEN
;  685     0747  5                  BEGIN
;  686     0748  5
;  687     0749  5                  FLAGS[SPACE_RECLAIMED] = 1;
;  688     0750  5
;  689     0751  5                  ! If the position of insertion of the new record follows the
;  690     0752  5                  ! current record in the bucket, then adjust it by the number
;  691     0753  5                  ! of bytes that were freed by the reformatting of the
;  692     0754  5                  ! current record.
;  693     0755  5                  !
;  694     0756  5                  IF .POS_INSERT GTRA .REC_ADDR
;  695     0757  5                  THEN
;  696     0758  5                      POS_INSERT = .POS_INSERT - .OLD_FREESPACE
;  697     0759  5                                               + .BKT_ADDR[BKT$W_FREESPACE];
;  698     0760  5                  END
;  699     0761  5
;  700     0762  5              ! If RMS is not able to reclaim any space from this RU modified
;  701     0763  5              ! record because it is locked by another stream, then proceed
;  702     0764  5              ! onto the next record in the primary data bucket.
;  703     0765  5              !
;  704     0766  4              ELSE
;  705     0767  4                  RM$GETNEXT_REC();
;  706     0768  4              END
;  707     0769  4
;  708     0770  4          ! If the current record in the bucket has not been marked as modified
;  709     0771  4          ! within a Recovery Unit but has been marked deleted, then completely
;  710     0772  4          ! recover its space, creating a RRV in its place (but at the end of the
;  711     0773  4          ! bucket) if necessary, and updating the bucket's freespace and the
;  712     0774  4          ! position of insertion of the new record as required. This can only be
;  713     0775  4          ! done if duplicate primary keys are not allowed, and of course, if the
;  714     0776  4          ! deleted record is not itself a deleted RRV.
;  715     0777  4          !
;  716     0778  3          ELSE
;  717     0779  3              IF  .REC_ADDR[IRC$V_DELETED]
;  718     0780  3                   AND
;  719     0781  3                  NOT .REC_ADDR[IRC$V_RRV]
;  720     0782  3                   AND
;  721     0783  3                  NOT .IDX_DFN[IDX$V_DUPKEYS]
;  722     0784  3              THEN
;  723     0785  4                  BEGIN
;  724     0786  4
;  725     0787  4                  LOCAL
;  726     0788  4                      NEXT_KEY_SIZE,
;  727     0789  4                      REC_OVHD,
;  728     0790  4                      REC_SIZE;
;  729     0791  4
;  730     0792  4                  ! Save the fact that a deleted record was encountered in this
;  731     0793  4                  ! primary data bucket and its space completely reclaimed.
;  732     0794  4                  !
;  733     0795  4                  FLAGS[SPACE_RECLAIMED] = 1;
;  734     0796  4
;  735     0797  4                  ! If the deleted record whose space is to be reclaimed preceeds
;  736     0798  4                  ! the point of insertion of the new record, then this position
```

```
;  737     0799  4                ! of insertion address must be adjusted, and it adjusted by two
;  738     0800  4                ! quantities.
;  739     0801  4                !
;  740     0802  4                ! 1. The number of bytes that are freed through the reclamation
;  741     0803  4                !    of the space occupied by the current record.
;  742     0804  4                !
;  743     0805  4                ! 2. If primary key compression is enabled and a record follows
;  744     0806  4                !    the current record, the number of bytes the key of this
;  745     0807  4                !    next record changes when its key is re-compressed as part
;  746     0808  4                !    of the removal of the current record.
;  747     0809  4                !
;  748     0810  4                IF .POS_INSERT GTRA .REC_ADDR
;  749     0811  4                THEN
;  750     0812  5                    BEGIN
;  751     0813  5
;  752     0814  5                    LOCAL
;  753     0815  5                        NEXT_REC_ADDR   : REF BBLOCK;
;  754     0816  5
;  755     0817  5                    REC_OVHD = RM$REC_OVHD(0; REC_SIZE);
;  756     0818  5                    NEXT_REC_ADDR = .REC_ADDR + .REC_OVHD + .REC_SIZE;
;  757     0819  5
;  758     0820  5                    ! Adjust the position of insertion of the new record by the
;  759     0821  5                    ! number of bytes which will be freed by the reclamation of
;  760     0822  5                    ! the current record.
;  761     0823  5                    !
;  762     0824  5                    POS_INSERT = .POS_INSERT - (.REC_OVHD + .REC_SIZE);
;  763     0825  5
;  764     0826  5                    ! If key compression is enabled, and there is a next record,
;  765     0827  5                    ! save the size of the key of the next record before it is
;  766     0828  5                    ! re-compressed as part of the deletion of the current
;  767     0829  5                    ! record. This size will be used to adjust the position of
;  768     0830  5                    ! insertion of the new record after the current record is
;  769     0831  5                    ! deleted and the key of the current record is
;  770     0832  5                    ! re-compressed.  However, don't adjust if POS_INSERT is
;  771     0833  5                    ! equal to REC_ADDR after the deleted record cleanup.
;  772     0834  5                    !
;  773     0835  5                    IF  .IDX_DFN[IDX$V_KEY_COMPR]
;  774     0836  5                        AND
;  775     0837  5                        .NEXT_REC_ADDR LSSA
;  776     0838  6                                       (.BKT_ADDR + .BKT_ADDR[BKT$W_FREESPACE])
;  777     0839  5                        AND
;  778     0840  5                        NOT .NEXT_REC_ADDR[IRC$V_RRV]
;  779     0841  5                        AND
;  780     0842  5                        .POS_INSERT GTRU .REC_ADDR      ! MUST still be true
;  781     0843  5                    THEN
;  782     0844  6                        BEGIN
;  783     0845  6                        FLAGS[KEY_EXPANSION] = 1;
;  784     0846  6                        NEXT_KEY_SIZE = .(.NEXT_REC_ADDR + .REC_OVHD)<0,8>
;  785     0847  5                        END;
;  786     0848  4                    END;
;  787     0849  4
;  788     0850  4                ! Recover the space occupied by the deleted record replacing it
;  789     0851  4                ! with an RRV at the end of the bucket if necessary, adjusting
;  790     0852  4                ! the bucket freespace offset, and re-compressing the key of
;  791     0853  4                ! the following record if primary key compression is enabled
;  792     0854  4                ! and there is a following record.
;  793     0855  4                !
```

```
;  794          0856  4                  RM$DELETE_UDR();
;  795          0857  4
;  796          0858  4                  ! If the address of the position of insertion of the new record
;  797          0859  4                  ! follows the address of the current record, and it is possible
;  798          0860  4                  ! that the size of the key of the following record might have
;  799          0861  4                  ! changed due to the re-compression of its primary key as part
;  800          0862  4                  ! of the reclamation of the space occupied by the current
;  801          0863  4                  ! record, then this possible change in key size must be used to
;  802          0864  4                  ! adjust the position of insertion of the new record.
;  803          0865  4                  !
;  804          0866  4                  IF TESTBITSC (FLAGS[KEY_EXPANSION])
;  805          0867  4                  THEN
;  806          0868  4                      POS_INSERT = .POS_INSERT + .(.REC_ADDR + .REC_OVHD)<0,8>
;  807          0869  4                                               - .NEXT_KEY_SIZE;
;  808          0870  4                  END
;  809          0871  4
;  810          0872  4              ! If the current record is neither marked deleted nor marked
;  811          0873  4              ! Recovery Unit modified then position to the next record.
;  812          0874  4              !
;  813          0875  3              ELSE
;  814          0876  3                  RM$GETNEXT_REC();
;  815          0877  2          END;
;  816          0878  2
;  817          0879  2      ! Readjust the offset to the point of insertion of the new record
;  818          0880  2      ! (regardless of whether this has or has not changed), restore into
;  819          0881  2      ! REC_ADDR the address of the point of insertion of the new record, and
;  820          0882  2      ! return whether RMS encountered any deleted records and recovered the
;  821          0883  2      ! space they occupied during its scan.
;  822          0884  2      !
;  823          0885  2      IRAB[IRB$W_POS_INS] = .POS_INSERT - .BKT_ADDR;
;  824          0886  2      REC_ADDR = .POS_INSERT;
;  825          0887  2      RETURN .FLAGS[SPACE_RECLAIMED];
;  826          0888  1      END;


                                        0C  BB 00000 RM$DEL_AND_TRY:
                                                     PUSHR   #^M<R2,R3>                      ; 0572
                                    5E  08  C2 00002         SUBL2   #8, SP                  ;
                                    09  1C  A7 E9 00005      BLBC    28(IDX_DFN), 1$         ; 0692
                              03    00A0  CA    01 E0 00009  BBS     #1, 160(IFAB), 1$       ; 0694
                                        00BE 31 0000F        BRW     12$                     ;
                                    04  AE  D4 00012 1$:     CLRL    FLAGS                   ; 0698
                                    52  56  D0 00015         MOVL    REC_ADDR, POS_INSERT    ; 0704
                                    56  0E  A5 9E 00018      MOVAB   14(R5), REC_ADDR        ; 0705
                                    50  04  A5 3C 0001C 2$:  MOVZWL  4(BKT_ADDR), R0         ; 0714
                              6E    55      50 C1 00020      ADDL3   R0, BKT_ADDR, (SP)      ;
                                    6E      56 D1 00024      CMPL    REC_ADDR, (SP)          ;
                                            03 1F 00027      BLSSU   4$                      ;
                                        0094 31 00029 3$:    BRW     11$                     ;
                              06        66      03 E1 0002C 4$:  BBC     #3, (REC_ADDR), 5$      ; 0716
                              F3    00A0  CA    01 E1 00030      BBC     #1, 160(IFAB), 3$       ; 0718
                              04        66      06 E0 00036 5$:  BBS     #6, (REC_ADDR), 6$      ; 0727
                              23        66      05 E1 0003A      BBC     #5, (REC_ADDR), 8$      ; 0729
                                    53  04  A5 B0 0003E 6$:     MOVW    4(BKT_ADDR), OLD_FREESPACE ; 0739
```

```
                                 0000G 30 00042          BSBW    RM$RU_RECLAIM                                   ; 0745
                              73   50 E9 00045          BLBC    R0, 10$                                         ;
                           04 AE   02 88 00048          BISB2   #2, FLAGS                                       ; 0749
                           56      52 D1 0004C          CMPL    POS_INSERT, REC_ADDR                            ; 0756
                                   CB 1B 0004F          BLEQU   2$                                              ;
                           50      53 3C 00051          MOVZWL  OLD_FREESPACE, R0                               ; 0758
                     50            50 C3 00054          SUBL3   R0, POS_INSERT, R0                              ;
                           52   04 A5 3C 00058          MOVZWL  4(BKT_ADDR), POS_INSERT                         ; 0759
                           52      50 C0 0005C          ADDL2   R0, POS_INSERT                                  ;
                                   BB 11 0005F 7$:      BRB     2$                                              ; 0745
                     56            02 E1 00061 8$:      BBC     #2, (REC_ADDR), 10$                             ; 0779
                     52            03 E0 00065          BBS     #3, (REC_ADDR), 10$                             ; 0781
                           4E   1C A7 E8 00069          BLBS    28(IDX_DFN), 10$                                ; 0783
                           04 AE   02 88 0006D          BISB2   #2, FLAGS                                       ; 0795
                           56      52 D1 00071          CMPL    POS_INSERT, REC_ADDR                            ; 0810
                                   30 1B 00074          BLEQU   9$                                              ;
                                   51 D4 00076          CLRL    R1                                              ; 0817
                                 0000G 30 00078          BSBW    RM$REC_OVHD                                     ;
                           53      50 D0 0007B          MOVL    R0, REC_OVHD                                    ;
                     50            53 C1 0007E          ADDL3   REC_OVHD, REC_ADDR, R0                          ; 0818
                           50      51 C0 00082          ADDL2   REC_SIZE, NEXT_REC_ADDR                         ;
                           51      53 C0 00085          ADDL2   REC_OVHD, R1                                    ; 0824
                           52      51 C2 00088          SUBL2   R1, POS_INSERT                                  ;
                     16    1C A7   06 E1 0008B          BBC     #6, 28(IDX_DFN), 9$                             ; 0835
                           6E      50 D1 00090          CMPL    NEXT_REC_ADDR, (SP)                             ; 0838
                                   11 1E 00093          BGEQU   9$                                              ;
                     0D            03 E0 00095          BBS     #3, (NEXT_REC_ADDR), 9$                         ; 0840
                           56      52 D1 00099          CMPL    POS_INSERT, REC_ADDR                            ; 0842
                                   08 1B 0009C          BLEQU   9$                                              ;
                           04 AE   01 88 0009E          BISB2   #1, FLAGS                                       ; 0845
                           6E    6340 9A 000A2          MOVZBL  (REC_OVHD)[NEXT_REC_ADDR], NEXT_KFY_SIZE        ; 0846
                                 0000G 30 000A6 9$:     BSBW    RM$DELETE_UDR                                   ; 0856
                     B1    04 AE   00 E5 000A9          BBCC    #0, FLAGS, 7$                                   ; 0866
                           50    6346 9A 000AE          MOVZBL  (REC_OVHD)[REC_ADDR], R0                        ; 0868
                           50      52 C0 000B2          ADDL2   POS_INSERT, R0                                  ;
                     52            6E C3 000B5          SUBL3   NEXT_KEY_SIZE, R0, POS_INSERT                   ; 0869
                                   A4 11 000B9          BRB     7$                                              ; 0779
                                 0000G 30 000BB 10$:    BSBW    RM$GETNEXT_REC                                  ; 0876
                                   9F 11 000BE          BRB     7$                                              ; 0714
                 48  A9            55 A3 000C0 11$:     SUBW3   BKT_ADDR, POS_INSERT, 72(IRAB)                  ; 0885
                           56      52 D0 000C5          MOVL    POS_INSERT, REC_ADDR                            ; 0886
           50    04 AE   01      01 EF 000C8          EXTZV   #1, #1, FLAGS, R0                               ; 0887
                                   02 11 000CE          BRB     13$                                             ;
                                   50 D4 000D0 12$:     CLRL    R0                                              ; 0888
                           5E      08 C0 000D2 13$:     ADDL2   #8, SP                                          ;
                                   0C BA 000D5          POPR    #^M<R2,R3>                                      ;
                                      05 000D7          RSB                                                     ;

; Routine Size:  216 bytes,    Routine Base:  RM$RMS3 + 00E7
```

```
;  828     0889  1 %SBTTL  'RM$INSERT_REC'
;  829     0890  1 GLOBAL ROUTINE RM$INSERT_REC(RECSZ) : RL$RABREG_4567 =
;  830     0891  1
;  831     0892  1 !++
;  832     0893  1 !
;  833     0894  1 ! FUNCTIONAL DESCRIPTION:
;  834     0895  1 !       routine to put the record into the bkt w/o any checks
;  835     0896  1 !
;  836     0897  1 ! CALLING SEQUENCE:
;  837     0898  1 !
;  838     0899  1 !       BSBW RM$INSERT_REC()
;  839     0900  1 !
;  840     0901  1 ! INPUT PARAMETERS:
;  841     0902  1 !       RECSZ - record size of record to be inserted including overhead
;  842     0903  1 !
;  843     0904  1 ! IMPLICIT INPUTS:
;  844     0905  1 !       BKT_ADDR, BDB of CURBDB
;  845     0906  1 !       IRAB -- POS_INS
;  846     0907  1 !       REC_ADDR -- pos of insert for record
;  847     0908  1 !
;  848     0909  1 ! OUTPUT PARAMETERS:
;  849     0910  1 !       NONE
;  850     0911  1 !
;  851     0912  1 ! IMPLICIT OUTPUTS:
;  852     0913  1 !       NONE
;  853     0914  1 !
;  854     0915  1 ! ROUTINE VALUE:
;  855     0916  1 !       success
;  856     0917  1 !
;  857     0918  1 ! SIDE EFFECTS:
;  858     0919  1 !       the bucket is expanded to make room for the record
;  859     0920  1 !       freespace is updated
;  860     0921  1 !       the bucket is marked valid and dirty
;  861     0922  1 !
;  862     0923  1 !--
;  863     0924  1
;  864     0925  2     BEGIN
;  865     0926  2
;  866     0927  2     EXTERNAL REGISTER
;  867     0928  2         COMMON_IO_STR,
;  868     0929  2         COMMON_RAB_STR,
;  869     0930  2         R_IDX_DFN_STR,
;  870     0931  2         R_REC_ADDR_STR;
;  871     0932  2
;  872     0933  2     ! The record will fit, get ready to move it in.
;  873     0934  2     !
;  874     0935  3     BEGIN
;  875     0936  3
;  876     0937  3     IF .BKT_ADDR[BKT$W_FREESPACE] NEQU .IRAB[IRB$W_POS_INS]
;  877     0938  3     THEN
;  878     0939  4         BEGIN
;  879     0940  4
;  880     0941  4         ! Since the record to be put is not the last one in the bucket, if
;  881     0942  4         ! keys are compressed, recompress the key of the next record, if it is
;  882     0943  4         ! not and RRV.  We are doing it for updates too, since when we deleted
;  883     0944  4         ! the record to be udpated, we expanded the key.
;  884     0945  4         !
```

```
;  885          0946  4          IF .IDX_DFN[IDX$V_KEY_COMPR]
;  886          0947  4            AND NOT .REC_ADDR[IRC$V_RRV]
;  887          0948  4          THEN
;  888          0949  4              RM$RECOMPR_KEY(.IRAB[IRB$L_RECBUF], .REC_ADDR + RM$REC_OVHD(0));
;  889          0950  4
;  890          0951  4          ! Since there is a hi set, move it down in the bucket to make room
;  891          0952  4          ! for the record.
;  892          0953  4          !
;  893          0954  4          RM$MOVE(.BKT_ADDR[BKT$W_FREESPACE] - .IRAB[IRB$W_POS_INS],
;  894          0955  4                  .REC_ADDR,
;  895          0956  4                  .REC_ADDR + .RECSZ);
;  896          0957  3          END;
;  897          0958  2      END;
;  898          0959  2
;  899          0960  3      BEGIN
;  900          0961  3
;  901          0962  3      ! update freespace word
;  902          0963  3      !
;  903          0964  3      BKT_ADDR[BKT$W_FREESPACE] = .BKT_ADDR[BKT$W_FREESPACE] + .RECSZ;
;  904          0965  3      BDB[BDB$V_DRT] = 1;
;  905          0966  3
;  906          0967  3      ! move new record into bucket
;  907          0968  3      !
;  908          0969  3      RETURN RM$BLDUDR(.RECSZ)                                                          ; R
;  909          0970  3
;  910          0971  3      END
;  911          0972  1      END;                                    ! { end of routine rm$insert_rec }


                       48   A9        04    A5  B1 00000 RM$INSERT_REC::
                                                 CMPW    4(BKT_ADDR), 72(IRAB)                              ; 0937
                                            31   13 00005          BEQL    2$                                 ;
                    14     1C   A7            06   E1 00007          BBC     #6, 28(IDX_DFN), 1$                ; 0946
                    10          66            03   E0 0000C          BBS     #3, (REC_ADDR), 1$                 ; 0947
                                              51   D4 00010          CLRL    R1                                 ; 0949
                                          0000G 30 00012          BSBW    RM$REC_OVHD                        ;
                    51          56            50   C1 00015          ADDL3   R0, REC_ADDR, R1                   ;
                                50        68   A9   D0 00019          MOVL    104(IRAB), R0                      ;
                                          0000G 30 0001D          BSBW    RM$RECOMPR_KEY                     ;
                                          04 BE46   9F 00020 1$:    PUSHAB  @RECSZ[REC_ADDR]                   ; 0956
                                             56   DD 00024          PUSHL   REC_ADDR                           ; 0955
                                50        04   A5   3C 00026          MOVZWL  4(BKT_ADDR), R0                    ; 0954
                                51        48   A9   3C 0002A          MOVZWL  72(IRAB), R1                       ;
                    7E          50             51   C3 0002E          SUBL3   R1, R0, -(SP)                      ;
                                          0000G 30 00032          BSBW    RM$MOVE                            ;
                                5E             0C   C0 00035          ADDL2   #12, SP                            ;
                       04       A5        04   AE   A0 00038 2$:     ADDW2   RECSZ, 4(BKT_ADDR)                 ; 0964
                       0A       A4             02   88 0003D          BISB2   #2, 10(BDB)                        ; 0965
                                          04   AE   DD 00041          PUSHL   RECSZ                              ; 0969
                                             FDFA 30 00044          BSBW    RM$BLDUDR                          ;
                                5E             04   C0 00047          ADDL2   #4, SP                             ;
                                                 05 0004A          RSB                                        ; 0972

; Routine Size:  75 bytes,    Routine Base:  RM$RMS3 + 01BF
```

```
;  912          0973  1
```

```
;  914          0974  1 %SBTTL  'RM$INSERT_UDR'
;  915          0975  1 GLOBAL ROUTINE RM$INSERT_UDR(RECSZ) : RL$RABREG_4567 =
;  916          0976  1
;  917          0977  1 !++
;  918          0978  1 !
;  919          0979  1 ! FUNCTIONAL DESCRIPTION:
;  920          0980  1 !
;  921          0981  1 !       Insert user data record in bucket, if possible
;  922          0982  1 !
;  923          0983  1 ! CALLING SEQUENCE:
;  924          0984  1 !
;  925          0985  1 !       BSBW RM$INSERT_LDR()
;  926          0986  1 !
;  927          0987  1 ! INPUT PARAMETERS:
;  928          0988  1 !       RECSZ - record size of record to be inserted including overhead
;  929          0989  1 !
;  930          0990  1 ! IMPLICIT INPUTS:
;  931          0991  1 !       RAB -- LOA bit, RSZ
;  932          0992  1 !       IDX_DFN -- DATBKTSIZ and DATFILL for bucket
;  933          0993  1 !       REC_ADDR -- pos of insert
;  934          0994  1 !       IFAB -- RFM of file
;  935          0995  1 !       IRAB -- CURBDB
;  936          0996  1 !       BDB and BKT_ADDR corresponding to CURBDB
;  937          0997  1 !                       from these we get the vbn, starting addr of bucket,
;  938          0998  1 !                       freespace pointer, NXTRECID, LSTRECID
;  939          0999  1 !
;  940          1000  1 ! OUTPUT PARAMETERS:
;  941          1001  1 !       RECSZ - record size of record to be inserted including overhead
;  942          1002  1 !
;  943          1003  1 ! IMPLICIT OUTPUTS:
;  944          1004  1 !       IRAB -- POS_INS
;  945          1005  1 !       BKT_ADDR -- NXTRECID and FREESPACE are updated
;  946          1006  1 !
;  947          1007  1 ! ROUTINE VALUE:
;  948          1008  1 !       SUC if record is successfully placed in bucket
;  949          1009  1 !       0 if record does not fit
;  950          1010  1 !
;  951          1011  1 ! SIDE EFFECTS:
;  952          1012  1 !       if it fits, record is placed into bucket
;  953          1013  1 !       and bucket is marked dirty and valid
;  954          1014  1 !
;  955          1015  1 !--
;  956          1016  1
;  957          1017  2     BEGIN
;  958          1018  2
;  959          1019  2     EXTERNAL REGISTER
;  960          1020  2         COMMON_IO_STR,
;  961          1021  2         R_IDX_DFN_STR,
;  962          1022  2         R_REC_ADDR_STR,
;  963          1023  2         COMMON_RAB_STR;
;  964          1024  2
;  965          1025  2     LOCAL
;  966          1026  2         REC_DEL,
;  967          1027  2         BKT_SIZE        : WORD;
;  968          1028  2
;  969          1029  2     MAP
;  970          1030  2         RECSZ           : REF VECTOR[1,LONG];
```

```
;  971          1031  2
;  972          1032  2      IRAB[IRB$W_POS_INS] = .REC_ADDR - .BKT_ADDR;
;  973          1033  2
;  974          1034  2      ! Set up bkt_size to be the fill size if loa set, else datbktsz * 512
;  975          1035  2      ! if the bkt is empty or all rrv's, use the whole bkt not the fill size
;  976          1036  2      ! if this is an update, use the whole bkt
;  977          1037  2      !
;  978          1038  2      BKT_SIZE = .IDX_DFN[IDX$B_DATBKTSZ]*512;
;  979          1039  2
;  980          1040  2      IF .RAB[RAB$V_LOA]
;  981          1041  2          AND
;  982          1042  2          NOT .IRAB[IRB$V_UPDATE]
;  983          1043  2      THEN
;  984          1044  3          BEGIN
;  985          1045  3
;  986          1046  3          LOCAL
;  987          1047  3              POINTER     : REF BBLOCK;
;  988          1048  3
;  989          1049  3          POINTER = .BKT_ADDR + BKT$C_OVERHDSZ;
;  990          1050  3
;  991          1051  3          IF .BKT_ADDR[BKT$W_FREESPACE] NEQU BKT$C_OVERHDSZ<0, 16>
;  992          1052  3              AND
;  993          1053  3              NOT .POINTER[IRC$V_RRV]
;  994          1054  3          THEN
;  995          1055  3              BKT_SIZE = .IDX_DFN[IDX$W_DATFILL];
;  996          1056  2          END;
;  997          1057  2
;  998          1058  2      IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  999          1059  2      THEN
; 1000          1060  2          BKT_SIZE = .BKT_SIZE - 1                        ! checksum byte
; 1001          1061  2      ELSE
; 1002          1062  2          BKT_SIZE = .BKT_SIZE - BKT$C_DATBKTOVH;
; 1003          1063  2
; 1004          1064  2      REC_DEL = 0;                            ! assume no record deleted
; 1005          1065  2
; 1006          1066  2      ! If freespace is already past usable space, or if rec size is
; 1007          1067  2      ! greater than usable space, won't fit
; 1008          1068  2      !
; 1009          1069  2      IF .BKT_ADDR [ BKT$W_FREESPACE ] GTRU .BKT_SIZE
; 1010          1070  3          OR .RECSZ [ 0 ] GTRU ( .BKT_SIZE - .BKT_ADDR [ BKT$W_FREESPACE ] )
; 1011          1071  2      THEN
; 1012          1072  2
; 1013          1073  2          ! Try to reclaim some space out of the bucket.  If we fail return zip!
; 1014          1074  2          !
; 1015          1075  3          IF NOT ( REC_DEL = RM$DEL_AND_TRY() )
; 1016          1076  2          THEN
; 1017          1077  2              RETURN 0;
; 1018          1078  2
; 1019          1079  2      ! If the key is compressed, and a record was deleted, it might have been
; 1020          1080  2      ! the one before the record.  So pack the record again to fix the key
; 1021          1081  2      ! compression.  Reset the last non-compressed record in case it was deleted.
; 1022          1082  2      !
; 1023          1083  2      IF .REC_DEL AND .IDX_DFN[IDX$V_KEY_COMPR]
; 1024          1084  2      THEN
; 1025          1085  3          BEGIN
; 1026          1086  3          IRAB[IRB$L_LST_NCMP] = .BKT_ADDR + BKT$C_OVERHDSZ;
; 1027          1087  3          RECSZ[0] = RM$PACK_REC();
```

```
;1028     1088  3          RECSZ[0] = .RECSZ[0] + IRC$C_FIXOVHSZ3;
;1029     1089  3
;1030     1090  3          IF .IFAB[IFB$B_RFMORG] NEQU FAB$C_FIX
;1031     1091  4            OR (.IFAB[IFB$B_RFMORG] EQL FAB$C_FIX
;1032     1092  4                AND .IDX_DFN[IDX$B_DATBKTYP] NEQU IDX$C_NCMPNCMP)
;1033     1093  3          THEN
;1034     1094  4              BEGIN
;1035     1095  4              RECSZ[0] = .RECSZ[0] + IRC$C_DATSZFLD;
;1036     1096  4
;1037     1097  4              ! If the state bit IRB$V_RU_UPDATE is set, then increase the record
;1038     1098  4              ! size by two to include the additional record size field which
;1039     1099  4              ! must be included within the record.
;1040     1100  4              !
;1041     1101  4              IF .IRAB[IRB$V_RU_UPDATE]
;1042     1102  4              THEN
;1043     1103  4                  RECSZ[0] = .RECSZ[0] + IRC$C_DATSZFLD;
;1044     1104  3              END;
;1045     1105  3
;1046     1106  2          END;
;1047     1107  2
;1048     1108  2      ! If the key compression changed, the record might have grown,
;1049     1109  2      ! make sure it still fits.
;1050     1110  2
;1051     1111  2      IF .BKT_ADDR[BKT$W_FREESPACE] GTRU .BKT_SIZE
;1052     1112  3        OR .RECSZ[0] GTRU ( .BKT_SIZE - .BKT_ADDR[BKT$W_FREESPACE] )
;1053     1113  2      THEN
;1054     1114  2          RETURN 0;
;1055     1115  2
;1056     1116  2      ! it's now o.k. to move the record in, so go do it
;1057     1117  2      !
;1058     1118  2      RETURN RM$INSERT_REC(.RECSZ[0]);
;1059     1119  2
;1060     1120  1      END;

                                0C  BB 00000 RM$INSERT_UDR::
                                             PUSHR   #^M<R2,R3>                          ;0975
48   A9           56            55  A3 00002         SUBW3   BKT_ADDR, REC_ADDR, 72(IRAB)        ;1032
                  50      17    A7  9A 00007         MOVZBL  23(IDX_DFN), R0                     ;1038
     52           50    0200    8F  A5 0000B         MULW3   #512, R0, BKT_SIZE
     17     05    A8            05  E1 00011         BBC     #5, 5(RAB), 1$                      ;1040
     12     06    A9            03  E0 00016         BBS     #3, 6(IRAB), 1$                     ;1042
                  50      0E    A5  9E 0001B         MOVAB   14(R5), POINTER                     ;1049
                  0E      04    A5  B1 0001F         CMPW    4(BKT_ADDR), #14                    ;1051
                                08  13 00023         BEQL    1$
     04           60            03  E0 00025         BBS     #3, (POINTER), 1$                   ;1053
                  52      26    A7  B0 00029         MOVW    38(IDX_DFN), BKT_SIZE               ;1055
                  03    00B7    CA  91 0002D 1$:     CMPB    183(IFAB), #3                       ;1058
                                04  1E 00032         BGEQU   2$
                                52  B7 00034         DECW    BKT_SIZE                            ;1060
                                03  11 00036         BRB     3$
                  52            02  A2 00038 2$:     SUBW2   #2, BKT_SIZE                        ;1062
                                50  D4 0003B 3$:     CLRL    REC_DEL                             ;1064
                  52      04    A5  B1 0003D         CMPW    4(BKT_ADDR), BKT_SIZE               ;1069
```

```
                                    10  1A 00041          BGTRU   4$                                          ;
                                51  52  3C 00043          MOVZWL  BKT_SIZE, R1                                ; 1070
                                53  04  A5 3C 00046       MOVZWL  4(BKT_ADDR), R3                             ;
                                51      53  C2 0004A      SUBL2   R3, RT                                      ;
                                51  0C  BE  D1 0004D      CMPL    @RECSZ, R1                                  ;
                                        08  1B 00051      BLEQU   5$                                          ;
                                    FE87    30 00053 4$:  BSBW    RM$DEL_AND_TRY                              ; 1075
                                05          50  E8 00056  BLBS    REC_DEL, 6$                                 ;
                                            53  11 00059  BRB     9$                                          ; 1077
                                2F          50  E9 0005B 5$:  BLBC    REC_DEL, 8$                           ; 1083
                            2A      1C  A7      06  E1 0005E 6$:  BBC     #6, 28(IDX_DFN), 8$                 ;
                                0098    C9  0E  A5  9E 00063  MOVAB   14(R5), 152(IRAB)                       ; 1086
                                        0000G   30  00069     BSBW    RM$PACK_REC                           ; 1087
                                    0C  BE      50  D0 0006C  MOVL    R0, @RECSZ                            ;
                                    0C  BE      09  C0 00070  ADDL2   #9, @RECSZ                            ; 1088
                                        01  50  AA  91 00074  CMPB    80(IFAB), #1                          ; 1090
                                            06  12 00078      BNEQ    7$                                    ;
                                        06  29  A7  91 0007A  CMPB    41(IDX_DFN), #6                       ; 1092
                                            0D  13 0007E      BEQL    8$                                    ;
                                    0C  BE      02  C0 00080 7$:  ADDL2   #2, @RECSZ                        ; 1095
                                            07  A9  95 00084  TSTB    7(IRAB)                               ; 1101
                                            04  18 00087      BGEQ    8$                                    ;
                                    0C  BE      02  C0 00089  ADDL2   #2, @RECSZ                            ; 1103
                                52      04  A5  B1 0008D 8$:  CMPW    4(BKT_ADDR), BKT_SIZE                 ; 1111
                                        1B  1A 00091      BGTRU   9$                                          ;
                                50          52  3C 00093  MOVZWL  BKT_SIZE, R0                             ; 1112
                                51      04  A5  3C 00096  MOVZWL  4(BKT_ADDR), R1                           ;
                                50          51  C2 0009A  SUBL2   R1, R0                                    ;
                                50      0C  BE  D1 0009D  CMPL    @RECSZ, R0                                ;
                                            0B  1A 000A1  BGTRU   9$                                        ;
                                        0C  BE  DD 000A3  PUSHL   @RECSZ                                    ; 1118
                                        FF0C    30 000A6  BSBW    RM$INSERT_REC                             ;
                                5E              04  C0 000A9  ADDL2   #4, SP                                ;
                                                02  11 000AC  BRB     10$                                    ;
                                                50  D4 000AE 9$:  CLRL    R0                                ; 1120
                                                0C  BA 000B0 10$: POPR    #^M<R2,R3>                        ;
                                                    05 000B2      RSB                                       ;

; Routine Size:  179 bytes,    Routine Base:  RM$RMS3 + 020A

; 1061          1121  1
; 1062          1122  1 END
; 1063          1123  1
; 1064          1124  0 ELUDOM


;                                   PSECT SUMMARY
;
;       Name                    Bytes                       Attributes
;
;  RM$RMS3                       701  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        71          2       154         00:00.4
```

```
;                                    COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3IUDR/OBJ=OBJ$:RM3IUDR MSRC$:RM3IUDR/UPDATE=(ENH$:RM3IUDR)

; Size:         701 code + 0 data bytes
; Run Time:       00:19.8
; Elapsed Time:   00:41.8
; Lines/CPU Min:   3412
; Lexemes/CPU-Min: 17234
; Memory Used:  143 pages
; Compilation Complete
```

RM3MAKIDX
LIS

RM3FNDRRV
LIS

RM3FNDRFA
LIS

RM3GET
LIS

RM3IUDR
LIS

RM3JOURNL
LIS

RM3KEYDSC
LIS

RM3MISC
LIS

RM3MISPUT
LIS