```
RRRRRRRRRRR      MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSS
RRR         RRR  MMMMMM    MMMMMM  SSS
RRR         RRR  MMMMMM    MMMMMM  SSS
RRR         RRR  MMMMMM    MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM   SSS
RRR         RRR  MMM   MMM   MMM   SSS
RRR         RRR  MMM   MMM   MMM   SSS
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  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
```

```
RRRRRRRR    MM      MM    333333    UU      UU  PPPPPPPP     SSSSSSSS    IIIIII    DDDDDDDD    XX      XX
RRRRRRRR    MM      MM    333333    UU      UU  PPPPPPPP     SSSSSSSS    IIIIII    DDDDDDDD    XX      XX
RR      RR  MMMM  MMMM  33      33  UU      UU  PP      PP  SS              II      DD      DD  XX      XX
RR      RR  MMMM  MMMM  33      33  UU      UU  PP      PP  SS              II      DD      DD  XX      XX
RR      RR  MM  MM  MM        33    UU      UU  PP      PP  SS              II      DD      DD    XX  XX
RR      RR  MM  MM  MM        33    UU      UU  PP      PP  SS              II      DD      DD    XX  XX
RRRRRRRR    MM      MM      33      UU      UU  PPPPPPPP     SSSSSS         II      DD      DD      XX
RRRRRRRR    MM      MM      33      UU      UU  PPPPPPPP     SSSSSS         II      DD      DD      XX
RR  RR      MM      MM        33    UU      UU  PP                    SS  II      DD      DD    XX  XX
RR  RR      MM      MM        33    UU      UU  PP                    SS  II      DD      DD    XX  XX
RR    RR    MM      MM  33      33  UU      UU  PP                    SS  II      DD      DD  XX      XX
RR    RR    MM      MM  33      33  UU      UU  PP                    SS  II      DD      DD  XX      XX
RR      RR  MM      MM    333333    UUUUUUUUUU  PP            SSSSSSSS    IIIIII    DDDDDDDD    XX      XX
RR      RR  MM      MM    333333    UUUUUUUUUU  PP            SSSSSSSS    IIIIII    DDDDDDDD    XX      XX

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 RM3UPSIDX (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 SIDR data record, all index updates
;    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:                       Christian Saether
;    45          0045  1 !
;    46          0046  1 ! CREATION DATE:                20-JUL-78  13:58
;    47          0047  1 !
;    48          0048  1 !
;    49          0049  1 ! MODIFIED BY:
;    50          0050  1 !
;    51          0051  1 !       V03-009 DGB0072         Donald G. Blair         24-Jul-1984
;    52          0052  1 !               During a root bucket split, the buckets are carefully
;    53          0053  1 !               written to disk in a certain order so as to minimize
;    54          0054  1 !               the possiblity of file corruption.  I needed to fix
;    55          0055  1 !               the error path so that buckets not yet written out
;    56          0056  1 !               to disk at the time of an error are marked as invalid
;    57          0057  1 !               so they aren't written out later to corrupt the file.
```

```
;    58          0058  1 ;
;    59          0059  1 ;       V03-008 MCN0003         Maria del C. Nasr       15-Mar-1983
;    60          0060  1 ;               More linkages reorganization
;    61          0061  1 ;
;    62          0062  1 ;       V03-007 MCN0002         Maria del C. Nasr       01-Mar-1983
;    63          0063  1 ;               Reorganize linkages
;    64          0064  1 ;
;    65          0065  1 ;       V03-006 TMK0004         Todd M. Katz            01-Feb-1983
;    66          0066  1 ;               Add support for Recovery Unit Journalling and RMS ROLLBACK
;    67          0067  1 ;               Recovery. When an attempt is made to insert a duplicate SIDR
;    68          0068  1 ;               into an index for a key of reference that does not allow
;    69          0069  1 ;               duplicates, before returning a duplicate key error determine
;    70          0070  1 ;               whether or not the last element in this SIDR array is marked
;    71          0071  1 ;               RU_DELETEd. It is only necessary to test the last SIDR array
;    72          0072  1 ;               element, because any SIDR array for a key of reference that
;    73          0073  1 ;               does not allow duplicates that is deleted within a Recovery Unit
;    74          0074  1 ;               is in effect "locked" by the stream doing the deletion for the
;    75          0075  1 ;               life of the Recovery Unit.
;    76          0076  1 ;
;    77          0077  1 ;               If the last SIDR element in the array is not marked RU_DELETE
;    78          0078  1 ;               then a duplicate key error is returned as before. Likewise, if
;    79          0079  1 ;               the last SIDR element is marked RU_DELETE but an attempt to
;    80          0080  1 ;               lock the corresponding primary data record fails because some
;    81          0081  1 ;               other process has it locked, then RMS concludes that the
;    82          0082  1 ;               Recovery Unit in which the element was deleted has not
;    83          0083  1 ;               concluded, and returns the duplicate key error.
;    84          0084  1 ;
;    85          0085  1 ;               However, if the last SIDR element in the array is marked
;    86          0086  1 ;               RU_DELETE and RMS is able to lock the SIDR, then RMS can
;    87          0087  1 ;               conclude that either it is the current stream that did the
;    88          0088  1 ;               delete within a Recovery Unit (in which case it already has the
;    89          0089  1 ;               entire SIDR array "locked"), or the Recovery Unit in which the
;    90          0090  1 ;               element was deleted (by some other process) has successfully
;    91          0091  1 ;               terminated. In either case RMS may proceed to insert the new
;    92          0092  1 ;               SIDR. In the latter case RMS reclaims the entire SIDR before
;    93          0093  1 ;               inserting the new SIDR, and of course, in the former case no
;    94          0094  1 ;               space reclamation is possible.
;    95          0095  1 ;
;    96          0096  1 ;       V03-005 TMK0003         Todd M. Katz            19-Sep-1981
;    97          0097  1 ;               Whenever key compression is enabled and a SIDR bucket is to be
;    98          0098  1 ;               updated, or index compression is enabled and an index bucket is
;    99          0099  1 ;               to be updated, the key of the new record (found in keybuffer 2)
;   100          0100  1 ;               is right-shifted two bytes to make room for the two key
;   101          0101  1 ;               compression overhead bytes, and those bytes are filled in. It
;   102          0102  1 ;               is also possible that a multi-bucket split occurring at the
;   103          0103  1 ;               primary data level will require the insertion of two new index
;   104          0104  1 ;               records into the level one index. The key of the second record
;   105          0105  1 ;               will be found in keybuffer 3, and it too should be shifted two
;   106          0106  1 ;               bytes and the key compression overhead bytes filled in
;   107          0107  1 ;               appropriately. This was not being done, and why everything
;   108          0108  1 ;               worked up to this point I don't know!
;   109          0109  1 ;
;   110          0110  1 ;       V03-004 TMK0002         Todd M. Katz            09-Sep-1981
;   111          0111  1 ;               The symbol IRB$B_SRCHFLAGS is now a word in size. Change all
;   112          0112  1 ;               references to it.
;   113          0113  1 ;
;   114          0114  1 ;               Add support for prologue SIDRs. This requires only a few minor
```

```
;  115     0115  1 !
;  116     0116  1 !               modifications to take into account the different structure of
;  117     0117  1 !               of prologue 3 SIDRs from prologue 1 and 2 SIDRs, and that their
;  118     0118  1 !               keys maybe compressed.
;  119     0119  1 !
;  120     0120  1 !       V03-003 KBT0237         Keith B. Thompson       23-Aug-1982
;  121     0121  1 !               Reorganize psects
;  122     0122  1 !
;  123     0123  1 !       V03-002 TMK0001         Todd M. Katz            02-Jul-1981
;  124     0124  1 !               Implement the RMS cluster solution for next record positioning.
;  125     0125  1 !               Since there is no longer a NRP list to update, do not bother
;  126     0126  1 !               to update it. In addition, since RMS will never squish out
;  127     0127  1 !               prologue 2 SIDR entries, never call the routine RM$RECVR_SPC
;  128     0128  1 !               (delete it) to reclaim SIDR space. Deleted entries will remain
;  129     0129  1 !               deleted for prologue 1 and 2.
;  130     0130  1 !
;  131     0131  1 !       V03-001 MCN0001         Maria del C. Nasr       25-Mar-1981
;  132     0132  1 !               Use macro to calculate key buffer address.
;  133     0133  1 !
;  134     0134  1 !       V018    TMK0001         Todd M. Katz    11-Feb-1982
;  135     0135  1 !               After an index bucket has been split, as part of the
;  136     0136  1 !               preparation for updating the index level immediatly above
;  137     0137  1 !               the current level, clear IRAB[IRB$L_VBN_MID]. There is a
;  138     0138  1 !               possibility that because a new index record must be inserted
;  139     0139  1 !               in the next level's index bucket, that index bucket may
;  140     0140  1 !               split. If the point of insertion of the new high key value
;  141     0141  1 !               resulting from the just split index bucket will be at the
;  142     0142  1 !               split point of the index bucket immediately above it, and
;  143     0143  1 !               if IRAB[IRB$L_VBN_MID] is not zero (which it won't be if a
;  144     0144  1 !               multibucket split occurred at the data level), the bucket
;  145     0145  1 !               at the next level may be incorrectly handled as a two-pass
;  146     0146  1 !               multibucket split instead of as a two-pass non-multibucket
;  147     0147  1 !               split. This will result in the corruption of the new index
;  148     0148  1 !               bucket. It will contain two identical keys with different
;  149     0149  1 !               VBN pointers, the low order key will have the same VBN
;  150     0150  1 !               pointer as the new high order key of the old bucket, and a
;  151     0151  1 !               pointer will be overwritten resulting in an inability to
;  152     0152  1 !               randomly access all records below it.
;  153     0153  1 !
;  154     0154  1 !       V017    CDS0001         C Saether       30-Aug-1981
;  155     0155  1 !               Reset CURBDB after release with keep lock, as
;  156     0156  1 !               it has changed and become the lock blb.
;  157     0157  1 !
;  158     0158  1 !       V016    PSK0003         P S Knibbe      09-Aug-1981
;  159     0159  1 !               Add support for splitting index buckets.
;  160     0160  1 !
;  161     0161  1 !       V015    PSK0002         P S Knibbe      29-Jul-1981
;  162     0162  1 !               Remove support for growing prologue three
;  163     0163  1 !               compressed indexes.
;  164     0164  1 !
;  165     0165  1 !       V014    PSK0001         P S Knibbe      14-Jun-1981
;  166     0166  1 !               Add support to RM$INS_IF_FIT for prologue three
;  167     0167  1 !               files.
;  168     0168  1 !               Add support to RM$INSS_OR_IDX for UKEY_ONLY
;  169     0169  1 !
;  170     0170  1 !       V013    CDS0081         C D Saether     26-Feb-1981     22:00
;  171     0171  1 !               Check for errors on split_em.
```

```
;  172     0172  1 !       V012    REFORMAT        D M Walp        24-JUL-1980
;  173     0173  1 !
;  174     0174  1 !       V011    CDS0080         C D Saether     27-FEB-1980
;  175     0175  1 !               Don't mark buffers invalid on errors.
;  176     0176  1 !
;  177     0177  1 !       V010    CDS0072         C D Saether     15-JAN-1980     14:50
;  178     0178  1 !               Don't zero or update nrp list unless splitting. (also
;  179     0179  1 !               corrects bug calling nrp routines with uninitialized value).
;  180     0180  1 !
;  181     0181  1 ! REVISION HISTORY:
;  182     0182  1 !
;  183     0183  1 !   Wendy Koenig,       12-OCT-78  14:51
;  184     0184  1 !   X0002 - CHANGE NRP STUFF
;  185     0185  1 !
;  186     0186  1 !   Wendy Koenig,       24-OCT-78  14:03
;  187     0187  1 !   X0003 - MAKE CHANGES CAUSED BY SHARING CONVENTIONS
;  188     0188  1 !
;  189     0189  1 !   Christian Saether,  12-DEC-78  20:40
;  190     0190  1 !   X0004 - handle case where SIDR pointer being added to deleted record
;  191     0191  1 !
;  192     0192  1 !   Christian Saether,  14-DEC-78  17:39
;  193     0193  1 !   X0005 - recvr_spc forces record to be deleted unless positioned for insert
;  194     0194  1 !           on it
;  195     0195  1 !
;  196     0196  1 !   Wendy Koenig,       25-JAN-79  11:26
;  197     0197  1 !   X0006 - GET RID OF SETTING VALID
;  198     0198  1 !
;  199     0199  1 !   Christian Saether,  1-july-79  11:30
;  200     0200  1 !   X0007 - set irb$v_dup when dupes seen on alternate
;  201     0201  1 !
;  202     0202  1 !   Christian Saether,  26-NOV-79  12:10
;  203     0203  1 !   0008 - don't force write thru if links don't change
;  204     0204  1 !
;  205     0205  1 !   Ron Schaefer,       11-JAN-80  16:50
;  206     0206  1 !   0009 - clear deleted-sidr flag on each call to RM$SQUISH_SIDR
;  207     0207  1 !
;  208     0208  1 !*****
;  209     0209  1
;  210     0210  1 LIBRARY 'RMSLIB:RMS';
;  211     0211  1
;  212     0212  1 REQUIRE 'RMSSRC:RMSIDXDEF';
;  213     0277  1
;  214     0278  1 ! Define default PSECTS for code.
;  215     0279  1 !
;  216     0280  1 PSECT
;  217     0281  1     CODE = RM$RMS3(PSECT_ATTR),
;  218     0282  1     PLIT = RM$RMS3(PSECT_ATTR);
;  219     0283  1
;  220     0284  1 ! linkages
;  221     0285  1 !
;  222     0286  1 LINKAGE
;  223     0287  1     L_PRESERVE1,
;  224     0288  1     L_QUERY_AND_LOCK,
;  225     0289  1     L_RABREG_4567,
;  226     0290  1     L_RABREG_567,
;  227     0291  1     L_RABREG_67,
;  228     0292  1     L_RABREG_7,
```

```
;  229     0293  1     L_RELEASE,
;  230     0294  1     L_SIDR_FIRST,
;  231     0295  1
;  232     0296  1     ! Local Linkage.
;  233     0297  1     !
;  234     0298  1     RL$INS_IF_FIT = JSB ()
;  235     0299  1                     : GLOBAL (R_BKT_ADDR, R_RAB, R_IRAB, R_IFAB, R_REC_ADDR,
;  236     0300  1                               R_IDX_DFN);
;  237     0301  1
;  238     0302  1 ! Forward Routines.
;  239     0303  1 !
;  240     0304  1 FORWARD ROUTINE
;  241     0305  1     RM$INS_IF_FIT       : RL$INS_IF_FIT;
;  242     0306  1
;  243     0307  1
;  244     0308  1 ! External Routines.
;  245     0309  1 !
;  246     0310  1 EXTERNAL ROUTINE
;  247     0311  1     RM$ALLOC_BKT        : RL$RABREG_7,
;  248     0312  1     RM$CSEARCH_TREE     : RL$RABREG_67,
;  249     0313  1     RM$EXT_ARRY_RFA     : RL$RABREG_67,
;  250     0314  1     RM$GETNXT_ARRAY     : RL$RABREG_67,
;  251     0315  1     RM$INS_REC          : RL$RABREG_67,
;  252     0316  1     RM$MOVE             : RL$PRESERVE1,
;  253     0317  1     RM$NEW_ROOT         : RL$RABREG_4567,
;  254     0318  1     RM$QUERY_PROC       : RL$QUERY_AND_LOCK ADDRESSING_MODE(GENERAL),
;  255     0319  1     RM$RECORD_SIZE      : RL$RABREG_567,
;  256     0320  1     RM$RLNERR           : RL$RELEASE ADDRESSING_MODE(LONG_RELATIVE),
;  257     0321  1     RM$RLSBKT           : RL$PRESERVE1,
;  258     0322  1     RM$SIDR_FIRST       : RL$SIDR_FIRST,
;  259     0323  1     RM$SQUISH_SIDR      : RL$RABREG_567,
;  260     0324  1     RM$SPLIT_EM         : RL$RABREG_67,
;  261     0325  1     RM$UPD_PLG          : RL$RABREG_7;
```

```
;  263     0326  1 %SBTTL 'RM$INSS_OR_IDX'
;  264     0327  1 GLOBAL ROUTINE RM$INSS_OR_IDX : RL$RABREG_567 =
;  265     0328  1
;  266     0329  1 !++
;  267     0330  1 !
;  268     0331  1 ! FUNCTIONAL DESCRIPTION:
;  269     0332  1 !       Call from level 0 to insert SIDR record and perform all neccessary
;  270     0333  1 !       index updates, or from level 1 on primary key to update index
;  271     0334  1 !
;  272     0335  1 ! CALLING SEQUENCE:
;  273     0336  1 !       RM$INSS_OR_IDX()
;  274     0337  1 !
;  275     0338  1 ! INPUT PARAMETERS:
;  276     0339  1 !       NONE
;  277     0340  1 !
;  278     0341  1 ! IMPLICIT INPUTS:
;  279     0342  1 !         IRAB -                pointer to internal RAB
;  280     0343  1 !            [ LOCK_BDB ] -     BDB of bucket to access if at level 1 on primary
;  281     0344  1 !                               key and LOCKABOVE used on position for insert
;  282     0345  1 !                               otherwise 0
;  283     0346  1 !            [ CURBDB ] -       locked BDB of level 0 if primary key. This is
;  284     0347  1 !                               released after successfully positioning at current
;  285     0348  1 !                               level 1.  For SIDR insert this is zero on entry
;  286     0349  1 !                               causing search down alternate index from root.
;  287     0350  1 !            [ STOPLEVEL ] - 1 for index update primary key, 0 for SIDR insert
;  288     0351  1 !            [ SPL_BITS ] -     status flags from primary data level split, 0 for
;  289     0352  1 !                               SIDR insert
;  290     0353  1 !            BIG_SPLIT -        more than two bucket split
;  291     0354  1 !            [ VBN_LEFT ] -     VBN of left hand bucket for primary key index
;  292     0355  1 !                               update
;  293     0356  1 !            [ VBN_RIGHT ] - VBN of right bkt prim key if present
;  294     0357  1 !            [ VBN_MID ] -      middle bkt VBN in 3-4 bkt prim key split case
;  295     0358  1 !            [ SRCHFLAGS ] - search flags for CSEARCH_TREE
;  296     0359  1 !            POSINSERT -        set to cause position for insert
;  297     0360  1 !            IDX_DFN -          pointer to index descriptor for key of reference
;  298     0361  1 !            [ DUPKEYS ] -      duplicate keys are allowed if set other fields as
;  299     0362  1 !                               used by routines called by this routine
;  300     0363  1 !
;  301     0364  1 ! OUTPUT PARAMETERS:
;  302     0365  1 !       NONE
;  303     0366  1 !
;  304     0367  1 ! IMPLICIT OUTPUTS:
;  305     0368  1 !       NONE
;  306     0369  1 !
;  307     0370  1 ! ROUTINE VALUE:
;  308     0371  1 !       SUC - success
;  309     0372  1 !       any error codes from allocation or get bucket routines
;  310     0373  1 !
;  311     0374  1 ! SIDE EFFECTS:
;  312     0375  1 !       NONE
;  313     0376  1 !
;  314     0377  1 !--
;  315     0378  1
;  316     0379  2     BEGIN
;  317     0380  2
;  318     0381  2     LITERAL
;  319     0382  2         TRUE = 1,
```

```
;  320          0383  2          FALSE = 0;
;  321          0384  2
;  322          0385  2      EXTERNAL REGISTER
;  323          0386  2          COMMON_RAB_STR,
;  324          0387  2          R_REC_ADDR_STR,
;  325          0388  2          R_IDX_DFN_STR,
;  326          0389  2          R_BKT_ADDR_STR;
;  327          0390  2
;  328          0391  2      GLOBAL REGISTER
;  329          0392  2          R_BDB_STR;
;  330          0393  2
;  331          0394  2      LOCAL
;  332          0395  2          ERRSTATUS,
;  333          0396  2          KILL_CUR;          ! Used only for error path -- true if we are to
;  334          0397  2                             ! throw away the updated contents of IRB$L_CURBDB;
;  335          0398  2                             ! false if we should write it to disk.
;  336          0399  2
;  337          0400  2      MACRO
;  338          0401  2
;  339        M 0402  2          EXONERR (CALL) =
;  340        M 0403  2          BEGIN
;  341        M 0404  2          IF NOT (ERRSTATUS = (CALL))
;  342        M 0405  2          THEN EXITLOOP
;  343          0406  2          END %,
;  344          0407  2
;  345          0408  2          ! This macro is used to handle errors after we have dirtied the
;  346          0409  2          ! bucket being split but before we have written it to disk.  In
;  347          0410  2          ! such cases, we want to throw away the dirty buffer.
;  348          0411  2          !
;  349        M 0412  2          EXONERR_KILL_CUR (CALL) =
;  350        M 0413  2          BEGIN
;  351        M 0414  2          IF NOT (ERRSTATUS = (CALL))
;  352        M 0415  2          THEN
;  353        M 0416  2              (KILL_CUR = TRUE;
;  354        M 0417  2              EXITLOOP)
;  355          0418  2          END %;
;  356          0419  2
;  357          0420  2      ! This routine is constructed as one while loop which is left via a return
;  358          0421  2      ! when no further index updates are neccessary
;  359          0422  2      !
;  360          0423  2
;  361          0424  2      WHILE 1
;  362          0425  2      DO
;  363          0426  3          BEGIN
;  364          0427  3
;  365          0428  3          ! By default, we save the curbdb contents on an error.
;  366          0429  3          KILL_CUR = FALSE;
;  367          0430  3
;  368          0431  3          ! if LOCK_BDB is nonzero then it was not released on the way down the
;  369          0432  3          ! tree and no further action is needed otherwise we must force a search
;  370          0433  3          ! from the root
;  371          0434  3          !
;  372          0435  3
;  373          0436  3          IF (BDB = .IRAB[IRB$L_LOCK_BDB]) NEQ 0
;  374          0437  3          THEN
;  375          0438  4              BEGIN
;  376          0439  4
```

```
;  377     0440  4            ! Swap current and lock bdb's and set up REC_ADDR.
;  378     0441  4            !
;  379     0442  4            REC_ADDR = .BDB[BDB$L_ADDR] + BKT$C_OVERHDSZ;
;  380     0443  4            IRAB[IRB$L_LOCK_BDB] = .IRAB[IRB$L_CURBDB];
;  381     0444  4            IRAB[IRB$L_CURBDB] = .BDB;
;  382     0445  4            END
;  383     0446  3        ELSE
;  384     0447  3
;  385     0448  3            ! Current bdb becomes lock bdb to be released later and curbdb is
;  386     0449  3            ! zeroed to force search from root.
;  387     0450  3            !
;  388     0451  4            BEGIN
;  389     0452  4            IRAB[IRB$L_LOCK_BDB] = .IRAB[IRB$L_CURBDB];
;  390     0453  4            IRAB[IRB$L_CURBDB] = 0;
;  391     0454  3            END;
;  392     0455  3
;  393     0456  3        EXONERR(RM$CSEARCH_TREE());
;  394     0457  3
;  395     0458  3        BKT_ADDR = .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$L_ADDR];
;  396     0459  3
;  397     0460  3        ! REC_ADDR is now pointing to the position of insert of the new record.
;  398     0461  3        ! If this is a prologue three bucket with compressed key records, then
;  399     0462  3        ! then shift the contents of keybuffer 2 down two bytes so that
;  400     0463  3        ! all key buffers look alike.
;  401     0464  3        !
;  402     0465  5        IF ((.BKT_ADDR[BKT$B_LEVEL] EQLU 0
;  403     0466  5                        AND
;  404     0467  5                    .IDX_DFN[IDX$V_KEY_COMPR])
;  405     0468  4             OR
;  406     0469  5            (.BKT_ADDR[BKT$B_LEVEL] NEQU 0
;  407     0470  5                        AND
;  408     0471  4                    .IDX_DFN[IDX$V_IDX_COMPR]))
;  409     0472  3        THEN
;  410     0473  4            BEGIN
;  411     0474  4
;  412     0475  4            MACRO
;  413     0476  4                KEYLEN          = 0,0,8,0 %,
;  414     0477  4                FRNT_CMPR       = 1,0,8,0 %;
;  415     0478  4
;  416     0479  4            LOCAL
;  417     0480  4                BUFF : REF BBLOCK;
;  418     0481  4
;  419     0482  4            BUFF = KEYBUF_ADDR(2);
;  420     0483  4            RM$MOVE (.IRAB[IRB$B_KEYSZ], .BUFF, .BUFF+2);
;  421     0484  4            BUFF [KEYLEN] = .IRAB [IRB$B_KEYSZ];
;  422     0485  4            BUFF [FRNT_CMPR] = 0;
;  423     0486  4
;  424     0487  4            ! If the level 1 index is to be updated with two index records
;  425     0488  4            ! because a multi-bucket split has taken place at the primary data
;  426     0489  4            ! record, then the key of the second index record (in keybuffer 3)
;  427     0490  4            ! should also be shifted down two bytes and the size and front
;  428     0491  4            ! compression count filled in so that all keybuffers continue to
;  429     0492  4            ! look alike.
;  430     0493  4            !
;  431     0494  4            IF .IRAB[IRB$V_BIG_SPLIT]
;  432     0495  4            THEN
;  433     0496  5                BEGIN
```

```
;  434     0497  5                BUFF = KEYBUF_ADDR(3);
;  435     0498  5                RM$MOVE (.IRAB[IRB$B_KEYSZ], .BUFF, .BUFF+2);
;  436     0499  5                BUFF [KEYLEN] = .IRAB [IRB$B_KEYSZ];
;  437     0500  5                BUFF [FRNT_CMPR] = 0;
;  438     0501  4                END;
;  439     0502  3            END;
;  440     0503  3
;  441     0504  3        ! If RMS is positioning to insert a SIDR and a duplicate was encountered
;  442     0505  3        ! during positioning then investigate further as to whether this does
;  443     0506  3        ! or doesn't represent an error.
;  444     0507  3        !
;  445     0508  3        IF .IRAB[IRB$B_STOPLEVEL] EQL 0
;  446     0509  3        THEN
;  447     0510  4            BEGIN
;  448     0511  4
;  449     0512  4            IF .IRAB[IRB$V_DUPS_SEEN]
;  450     0513  4            THEN
;  451     0514  4
;  452     0515  4                ! If duplicates were seen and this key of reference does not
;  453     0516  4                ! allow duplicate keys then this will represent an error unless
;  454     0517  4                ! all the elements in the array were deleted within a Recovery
;  455     0518  4                ! Unit that has since terminated successfully or by the current
;  456     0519  4                ! stream whose process is still within a Recovery Unit.
;  457     0520  4                !
;  458     0521  4                IF NOT .IDX_DFN[IDX$V_DUPKEYS]
;  459     0522  4                THEN
;  460     0523  5                    BEGIN
;  461     0524  5
;  462     0525  5                    LOCAL
;  463     0526  5                        BEG_OF_SIDR,
;  464     0527  5                        END_OF_SIDR,
;  465     0528  5                        LAST_SIDR : REF BBLOCK;
;  466     0529  5
;  467     0530  5                    ! Position to the last element in the current SIDR array.
;  468     0531  5                    ! It is only necessary to determine the status of this
;  469     0532  5                    ! element in order to determine whether or not the
;  470     0533  5                    ! insertion of this duplicate represents an error or not.
;  471     0534  5                    !
;  472     0535  5                    END_OF_SIDR = .REC_ADDR;
;  473     0536  5                    REC_ADDR    = .IRAB[IRB$L_LST_REC];
;  474     0537  5                    BEG_OF_SIDR = .REC_ADDR;
;  475     0538  5                    REC_ADDR    = RM$SIDR_FIRST (0);
;  476     0539  5
;  477     0540  5                    DO
;  478     0541  6                        BEGIN
;  479     0542  6                        LAST_SIDR = .REC_ADDR;
;  480     0543  6                        RM$GETNXT_ARRAY();
;  481     0544  6                        END
;  482     0545  5                    UNTIL .REC_ADDR GEQA .END_OF_SIDR;
;  483     0546  5
;  484     0547  5                    ! If the last element in the current SIDR array was deleted
;  485     0548  5                    ! within a Recovery Unit, then RMS may still be able to
;  486     0549  5                    ! insert this new element provided it would be able to
;  487     0550  5                    ! lock the primary data record the SIDR element points to.
;  488     0551  5                    ! Being able to lock the record will indicate either that
;  489     0552  5                    ! the Recovery Unit in which the SIDR element was deleted
;  490     0553  5                    ! has successfully terminated, or that it was the current
```

```
;  491     0554  5                    ! stream that deleted the element within a Recovery Unit
;  492     0555  5                    ! (which may still be active). In either case, RMS may
;  493     0556  5                    ! consider the element to be deleted, and as all elements
;  494     0557  5                    ! in the array would then be deleted, RMS can proceed to
;  495     0558  5                    ! insert the new element.
;  496     0559  5                    !
;  497     0560  5                    IF .LAST_SIDR[IRC$V_RU_DELETE]
;  498     0561  5                    THEN
;  499     0562  6                        BEGIN
;  500     0563  6
;  501     0564  6                        LOCAL
;  502     0565  6                            ID,
;  503     0566  6                            TEMP_STATUS,
;  504     0567  6                            VBN;
;  505     0568  6
;  506     0569  6                        ! Extract the RFA out of the SIDR and determine without
;  507     0570  6                        ! waiting the lock status of the corresponding primary
;  508     0571  6                        ! data record.
;  509     0572  6                        !
;  510     0573  6                        REC_ADDR = .LAST_SIDR;
;  511     0574  6                        RM$EXT_ARRY_RFA (VBN, ID);
;  512     0575  6                        IRAB[IRB$V_NO_Q_WAIT] = 1;
;  513     0576  6
;  514     0577  6                        ! If RMS is able to lock the corresponding primary data
;  515     0578  6                        ! record, then it may treat the SIDR array element as
;  516     0579  6                        ! being deleted and proceed to insert the new element.
;  517     0580  6                        !
;  518     0581  7                        IF (TEMP_STATUS = RM$QUERY_PROC (.VBN, .ID))
;  519     0582  6                        THEN
;  520     0583  6
;  521     0584  6                            ! If the last element was deleted within another
;  522     0585  6                            ! process's Recovery Unit, or if the current
;  523     0586  6                            ! process is not in a Recovery Unit, then RMS may
;  524     0587  6                            ! perform some space reclamation before inserting
;  525     0588  6                            ! the new element. Space reclamation will consist
;  526     0589  6                            ! of deleting the entire SIDR array.
;  527     0590  6                            !
;  528     0591  7                            IF  .TEMP_STATUS<0,16> EQLU RMSSUC()
;  529     0592  6                                OR
;  530     0593  6                                NOT .IFAB[IFB$V_RUP]
;  531     0594  6                            THEN
;  532     0595  7                                BEGIN
;  533     0596  7                                RM$SQUISH_SIDR (0, .BEG_OF_SIDR);
;  534     0597  7                                IRAB[IRB$V_DUPS_SEEN] = 0;
;  535     0598  7                                IRAB[IRB$V_DUP_KEY] = 0;
;  536     0599  7                                END
;  537     0600  7
;  538     0601  7                            ! Otherwise, RMS can not perform any space
;  539     0602  7                            ! reclamation, and instead positions to the
;  540     0603  7                            ! insertion point of the new element.
;  541     0604  7                            !
;  542     0605  6                            ELSE
;  543     0606  6                                RM$GETNXT_ARRAY()
;  544     0607  6
;  545     0608  6                    ! If RMS is not able to lock the primary data record
;  546     0609  6                    ! that the last SIDR element points to then RMS can not
;  547     0610  6                    ! consider all the elements in the SIDR array to be
```

```
;  548     0611  6                    ! deleted. In this case RMS can not insert this new
;  549     0612  6                    ! new element, but instead returns a duplicate key
;  550     0613  6                    ! error. If RMS were to insert the SIDR and the
;  551     0614  6                    ! Recovery Unit failed, then after Recovery Unit
;  552     0615  6                    ! ROLLBACK this SIDR array would have two non-deleted
;  553     0616  6                    ! elements even though this key of reference does not
;  554     0617  6                    ! allow duplicates.
;  555     0618  6                    !
;  556     0619  6                    ELSE
;  557     0620  6                        ERRSTATUS = RMSERR(DUP);
;  558     0621  6                    END
;  559     0622  6
;  560     0623  6                ! If the last element in the current SIDR array was not
;  561     0624  6                ! deleted within a Recovery Unit, then RMS can not insert
;  562     0625  6                ! this new element and instead must return a duplicate key
;  563     0626  6                ! error.
;  564     0627  6                !
;  565     0628  5                ELSE
;  566     0629  5                    ERRSTATUS = RMSERR(DUP);
;  567     0630  5
;  568     0631  6                IF .ERRSTATUS<0,16> EQLU RMSERR(DUP)
;  569     0632  5                THEN
;  570     0633  5                    EXITLOOP;
;  571     0634  5                END
;  572     0635  5
;  573     0636  5            ! As this key of reference allows duplicate keys, and a
;  574     0637  5            ! duplicate was seen, save that information so that the proper
;  575     0638  5            ! success status may eventually be returned.
;  576     0639  5            !
;  577     0640  4            ELSE
;  578     0641  4                IRAB[IRB$V_DUP] = 1;
;  579     0642  4
;  580     0643  4            END
;  581     0644  4
;  582     0645  4        ! If this wasn't position to level 0 then release lock on level
;  583     0646  4        ! below after positioning to point of insert above.
;  584     0647  4        !
;  585     0648  3        ELSE
;  586     0649  3            RELEASE(IRAB[IRB$L_LOCK_BDB]);
;  587     0650  3
;  588     0651  3        BDB = .IRAB[IRB$L_CURBDB];
;  589     0652  3        BDB[BDB$V_DRT] = 1;
;  590     0653  3
;  591     0654  3        ! Now try to put the record into the existing bucket - success if it
;  592     0655  3        ! fits.
;  593     0656  3        !
;  594     0657  3        IF RM$INS_IF_FIT()
;  595     0658  3        THEN
;  596     0659  4            BEGIN
;  597     0660  4
;  598     0661  4            ! Record fits without splitting so release lock bdb (there is
;  599     0662  4            ! one only at level 0 when lock above was used on positioning)
;  600     0663  4            ! write thru bucket and return.
;  601     0664  4            !
;  602     0665  4            LOCAL
;  603     0666  4                FLAGS;
;  604     0667  4
```

```
;  605     0668  4              FLAGS = 0;
;  606     0669  4
;  607     0670  4              IF (BDB = .IRAB[IRB$L_LOCK_BDB]) NEQ 0
;  608     0671  4              THEN
;  609     0672  5                  BEGIN
;  610     0673  5                  IRAB[IRB$L_LOCK_BDB] = 0;
;  611     0674  5                  RM$RLSBKT(0);
;  612     0675  4                  END;
;  613     0676  4
;  614     0677  4              BDB = .IRAB[IRB$L_CURBDB];
;  615     0678  4              IRAB[IRB$L_CURBDB] = 0;
;  616     0679  4              RETURN RM$RLSBKT(.FLAGS);
;  617     0680  4
;  618     0681  3              END;
;  619     0682  3
;  620     0683  3          ! Allocate a new bucket to split into.
;  621     0684  3          !
;  622     0685  3          EXONERR(RM$ALLOC_BKT());
;  623     0686  3
;  624     0687  3          ! If LOCKABOVE was used and we are doing a SIDR data level split there
;  625     0688  3          ! are now 3 buffers in use.
;  626     0689  3          !
;  627     0690  3          BDB = .IRAB[IRB$L_NXTBDB];
;  628     0691  3
;  629     0692  3          ! Split the bucket !!!
;  630     0693  3          !
;  631     0694  4          IF NOT (ERRSTATUS = RM$SPLIT_EM())
;  632     0695  3          THEN
;  633     0696  4              BEGIN
;  634     0697  4              BDB [BDB$V_VAL] = 0;
;  635     0698  4              IRAB [IRB$L_NXTBDB] = 0;
;  636     0699  4              RM$RLSBKT(0);
;  637     0700  4              BBLOCK [.IRAB[IRB$L_CURBDB], BDB$V_VAL] = 0;
;  638     0701  4              EXITLOOP
;  639     0702  3              END;
;  640     0703  3
;  641     0704  3          ! Now save the VBN of the new bucket for next level update.
;  642     0705  3          !
;  643     0706  3          IRAB[IRB$L_VBN_RIGHT] = .BDB[BDB$L_VBN];
;  644     0707  3          BDB[BDB$V_DRT] = 1;
;  645     0708  3          IRAB[IRB$L_NXTBDB] = 0;
;  646     0709  3
;  647     0710  3          ! We must clear VBN_MID for the next level update as a precaution.
;  648     0711  3          ! If the current index bucket split was for a multibucket data level
;  649     0712  3          ! split, the update at the next level could be done incorrectly if
;  650     0713  3          ! that index bucket split and the point of insertion of the new key
;  651     0714  3          ! was at the split point, and if this VBN cell is not zero.
;  652     0715  3          !
;  653     0716  3          IRAB[IRB$L_VBN_MID] = 0;
;  654     0717  3
;  655     0718  3          ! Write the new bucket.
;  656     0719  3          !
;  657     0720  3          EXONERR_KILL_CUR(RM$RLSBKT(RLS$M_WRT_THRU));
;  658     0721  3
;  659     0722  3          ! If this was a continuation bucket then no index update is neccessary
;  660     0723  3          ! so release lock bdb if any and write out current bdb.
;  661     0724  3          !
```

```
;  662     0725  3          IF .IRAB[IRB$V_CONT_BKT]
;  663     0726  3          THEN
;  664     0727  4              BEGIN
;  665     0728  4
;  666     0729  4              LOCAL
;  667     0730  4                  FLAGS;
;  668     0731  4
;  669     0732  4              FLAGS = RLS$M_WRT_THRU;
;  670     0733  4
;  671     0734  4              IF (BDB = .IRAB[IRB$L_LOCK_BDB]) NEQ 0
;  672     0735  4              THEN
;  673     0736  5                  (IRAB[IRB$L_LOCK_BDB] = 0;
;  674     0737  4                  RM$RLSBKT(0));
;  675     0738  4
;  676     0739  4              BDB = .IRAB[IRB$L_CURBDB];
;  677     0740  4              IRAB[IRB$L_CURBDB] = 0;
;  678     0741  4              RETURN RM$RLSBKT(.FLAGS);
;  679     0742  4
;  680     0743  3              END;
;  681     0744  3
;  682     0745  3          ! Set up BDB and BKT_ADDR for new root code if taken or releasing
;  683     0746  3          ! CURBDB if not and VBN_LEFT for next pass, i.e., the index update or
;  684     0747  3          ! new root generation.
;  685     0748  3          !
;  686     0749  3          BDB = .IRAB[IRB$L_CURBDB];
;  687     0750  3          BKT_ADDR = .BDB[BDB$L_ADDR];
;  688     0751  3          IRAB[IRB$L_VBN_LEFT] = .BDB[BDB$L_VBN];
;  689     0752  3
;  690     0753  3          IF .BKT_ADDR[BKT$V_ROOTBKT]
;  691     0754  3              AND
;  692     0755  3              .BKT_ADDR[BKT$B_LEVEL] EQL .IDX_DFN[IDX$B_ROOTLEV]
;  693     0756  3          THEN
;  694     0757  3
;  695     0758  3              ! This is a root bucket which split so link in new bucket make new
;  696     0759  3              ! root, make non root out of old bucket.
;  697     0760  3              !
;  698     0761  4              BEGIN
;  699     0762  4              BKT_ADDR[BKT$V_ROOTBKT] = 0;
;  700     0763  4              EXONERR_KILL_CUR(RM$ALLOC_BKT());
;  701     0764  4
;  702     0765  4              ! Restore next bucket link of original bucket that got clobbered
;  703     0766  4              ! when we linked in a bucket for the new root.
;  704     0767  4              !
;  705     0768  4              BKT_ADDR[BKT$L_NXTBKT] = .IRAB[IRB$L_VBN_RIGHT];
;  706     0769  4
;  707     0770  4              ! Set up BDB and BKT_ADDR for NEW_ROOT.
;  708     0771  4              !
;  709     0772  4              BDB = .IRAB[IRB$L_NXTBDB];
;  710     0773  4              BKT_ADDR = .BDB[BDB$L_ADDR];
;  711     0774  4              RM$NEW_ROOT();
;  712     0775  4
;  713     0776  4              ! Write out and release new root.
;  714     0777  4              !
;  715     0778  4              BDB[BDB$V_DRT] = 1;
;  716     0779  4              IRAB[IRB$L_NXTBDB] = 0;
;  717     0780  4              EXONERR_KILL_CUR(RM$RLSBKT(RLS$M_WRT_THRU));
;  718     0781  4
```

```
;  719          0782  4              ! Update all relevant prologue information.
;  720          0783  4              !
;  721          0784  4              EXONERR_KILL_CUR(RM$UPD_PLG());
;  722          0785  4
;  723          0786  4              ! Now write out original root bucket.
;  724          0787  4              !
;  725          0788  4              BDB = .IRAB[IRB$L_CURBDB];
;  726          0789  4              IRAB[IRB$L_CURBDB] = 0;
;  727          0790  4              RETURN (RM$RLSBKT(RLS$M_WRT_THRU));
;  728          0791  4
;  729          0792  3              END;
;  730          0793  3
;  731          0794  3          ! Write out current BDB keeping lock on it until positioned to level
;  732          0795  3          ! above on index update.
;  733          0796  3          !
;  734        P 0797  3          EXONERR(RM$RLSBKT(RLS$M_WRT_THRU
;  735        P 0798  3                  OR
;  736          0799  3                  RLS$M_KEEP_LOCK));
;  737          0800  3          IRAB[IRB$L_CURBDB] = .BDB;
;  738          0801  3
;  739          0802  3          IRAB[IRB$B_STOPLEVEL] = .IRAB[IRB$B_STOPLEVEL] + 1;
;  740          0803  3          IRAB[IRB$W_SRCHFLAGS] = IRB$M_POSINSERT;
;  741          0804  3          IRAB[IRB$B_SPL_BITS] = 0;
;  742          0805  2          END;    ! end of WHILE loop
;  743          0806  2
;  744          0807  2      ! This is the error code to release LOCK_BDB and CURBDB if they
;  745          0808  2      ! are non-zero. This code is only executed on errors.
;  746          0809  2      !
;  747          0810  2      IF (BDB = .IRAB[IRB$L_LOCK_BDB]) NEQ 0
;  748          0811  2      THEN
;  749          0812  3          (IRAB[IRB$L_LOCK_BDB] = 0;
;  750          0813  2          RM$RLSBKT(0));
;  751          0814  2
;  752          0815  2      ! If .kill_cur is true, we call release-no-error to pitch the
;  753          0816  2      ! dirty contents of the curbdb buffer, else we call release-bucket
;  754          0817  2      ! to release the buffer but save the contents.
;  755          0818  2      !
;  756          0819  2      IF (BDB = .IRAB[IRB$L_CURBDB]) NEQ 0
;  757          0820  2      THEN
;  758          0821  3          BEGIN
;  759          0822  3          IRAB[IRB$L_CURBDB] = 0;
;  760          0823  3          IF .KILL_CUR THEN
;  761          0824  3              RM$RLSNERR(0)
;  762          0825  3          ELSE
;  763          0826  3              RM$RLSBKT(0);
;  764          0827  2          END;
;  765          0828  2
;  766          0829  2      RETURN .ERRSTATUS;
;  767          0830  2
;  768          0831  1      END;


                                                    .TITLE  RM3UPSIDX
                                                    .IDENT  \V04-000\

                                                    .EXTRN  RM$ALLOC_BKT, RM$CSEARCH_TREE
                                                    .EXTRN  RM$EXT_ARRY_RFA
```

```
                                        .EXTRN  RM$GETNXT_ARRAY
                                        .EXTRN  RM$INS_REC, RM$MOVE
                                        .EXTRN  RM$NEW_ROOT, RM$QUERY_PROC
                                        .EXTRN  RM$RECORD_SIZE, RM$RLNERR
                                        .EXTRN  RM$RLSBKT, RM$SIDR_FIRST
                                        .EXTRN  RM$SQUISH_SIDR, RM$SPLIT_EM
                                        .EXTRN  RM$UPD_PLG

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

                                1C  BB 00000 RM$INSS_OR_IDX::
                                            PUSHR   #^M<R2,R3,R4>                       ; 0327
                             5E         18  C2 00002            SUBL2   #24, SP
                          08 AE     20    A9  9E 00005            MOVAB   32(R9), 8(SP)       ; 0443
                                            6E  D4 0000A 1$:        CLRL    KILL_CUR            ; 0429
                             50   0084    C9  9E 0000C            MOVAB   132(IRAB), R0       ; 0436
                             54           60  D0 00011            MOVL    (R0), BDB           ;
                                          0F  13 00014            BEQL    2$                  ;
                       56 18 A4           0E  C1 00016            ADDL3   #14, 24(BDB), REC_ADDR  ; 0442
                             60     08    BE  D0 0001B            MOVL    @8(SP), (R0)        ; 0443
                          08 BE           54  D0 0001F            MOVL    BDB, @8(SP)         ; 0444
                                          07  11 00023            BRB     3$                  ; 0436
                             60     08    BE  D0 00025 2$:        MOVL    @8(SP), (R0)        ; 0452
                                    08    BE  D4 00029            CLRL    @8(SP)              ; 0453
                                      0000G 30  0002C 3$:        BSBW    RM$CSEARCH_TREE     ; 0456
                          04 AE             50  D0 0002F            MOVL    R0, ERRSTATUS       ;
                                    03      04  AE  E8 00033        BLBS    ERRSTATUS, 4$       ;
                                          022E  31  00037           BRW     29$                 ;
                             50     20      A9  D0 0003A 4$:        MOVL    32(IRAB), R0        ; 0458
                             55     18      A0  D0 0003E            MOVL    24(R0), BKT_ADDR    ;
                                    0C      A5  95 00042            TSTB    12(BKT_ADDR)        ; 0465
                                            07  12 00045            BNEQ    5$                  ;
                    07 1C A7                06  E0 00047            BBS     #6, 28(IDX_DFN), 6$ ; 0467
                                            47  13 0004C            BEQL    7$                  ; 0469
                    42 1C A7                03  E1 0004E 5$:        BBC     #3, 28(IDX_DFN), 7$ ; 0471
                             51     00B4    CA  3C 00053 6$:        MOVZWL  180(IFAB), BUFF     ; 0482
                             51     60      A9  C0 00058            ADDL2   96(IRAB), BUFF      ;
                                    02      A1  9F 0005C            PUSHAB  2(BUFF)             ; 0483
                                            51  DD 0005F            PUSHL   BUFF                ;
                             7E     00A6    C9  9A 00061            MOVZBL  166(IRAB), -(SP)    ;
                                      0000G 30  00066           BSBW    RM$MOVE             ;
                             5E             0C  C0 00069            ADDL2   #12, SP             ;
                             61     00A6    C9  9B 0006C            MOVZBW  166(IRAB), (BUFF)   ; 0484
                    1F 44 A9                02  E1 00071            BBC     #2, 68(IRAB), 7$    ; 0494
                             50     00B4    CA  3C 00076            MOVZWL  180(IFAB), R0       ; 0497
                             51     60 B940 3E  0007B            MOVAW   @96(IRAB)[R0], BUFF ;
                                    02      A1  9F 00080            PUSHAB  2(BUFF)             ; 0498
                                            51  DD 00083            PUSHL   BUFF                ;
                             7E     00A6    C9  9A 00085            MOVZBL  166(IRAB), -(SP)    ;
                                      0000G 30  0008A           BSBW    RM$MOVE             ;
                             5E             0C  C0 0008D            ADDL2   #12, SP             ;
                             61     00A6    C9  9B 00090            MOVZBW  166(IRAB), (BUFF)   ; 0499
                                    41      A9  95 00095 7$:        TSTB    65(IRAB)            ; 0508
                                            03  13 00098            BEQL    8$                  ;
                                          008E  31  0009A           BRW     16$                 ;
                                    44      A9  95 0009D 8$:        TSTB    68(IRAB)            ; 0512
                                            03  19 000A0            BLSS    9$                  ;
```

```
                              0097  31 000A2          BRW     17$                              ;
            7C          1C      A7  E8 000A5  9$:     BLBS    28(IDX_DFN), 15$                 ; 0521
            53                  56  D0 000A9          MOVL    REC_ADDR, END_OF_SIDR            ; 0535
            56          4C      A9  D0 000AC          MOVL    76(IRAB), REC_ADDR               ; 0536
      0C    AE                  56  D0 000B0          MOVL    REC_ADDR, BEG_OF_SIDR            ; 0537
                                7E  D4 000B4          CLRL    -(SP)                            ; 0538
                             0000G  30 000B6          BSBW    RM$SIDR_FIRST                    ;
            5E                  04  C0 000B9          ADDL2   #4, SP                           ;
            56                  50  D0 000BC          MOVL    R0, REC_ADDR                     ;
            52                  56  D0 000BF  10$:    MOVL    REC_ADDR, LAST_SIDR              ; 0542
                             0000G  30 000C2          BSBW    RM$GETNXT_ARRAY                  ; 0543
            53                  56  D1 000C5          CMPL    REC_ADDR, END_OF_SIDR            ; 0545
                                F5  1F 000C8          BLSSU   10$                              ;
   47       62                  05  E1 000CA          BBC     #5, (LAST_SIDR), 13$             ; 0560
            56                  52  D0 000CE          MOVL    LAST_SIDR, REC_ADDR              ; 0573
                        10      AE  9F 000D1          PUSHAB  ID                               ; 0574
                        18      AE  9F 000D4          PUSHAB  VBN                              ;
                             0000G  30 000D7          BSBW    RM$EXT_ARRY_RFA                  ;
            5E                  08  C0 000DA          ADDL2   #8, SP                           ;
      07    A9                  01  88 000DD          BISB2   #1, 7(IRAB)                      ; 0575
            52          10      AE  D0 000E1          MOVL    ID, R2                           ; 0581
            51          14      AE  D0 000E5          MOVL    VBN, R1                          ;
                    00000000G   00  16 000E9          JSB     RM$QUERY_PROC                    ;
            23                  50  E9 000EF          BLBC    TEMP_STATUS, 13$                 ;
            01                  50  B1 000F2          CMPW    TEMP_STATUS, #1                  ; 0591
                                06  13 000F5          BEQL    11$                              ;
   13  00A2 CA                  02  E0 000F7          BBS     #2, 162(IFAB), 12$               ; 0593
                        0C      AE  DD 000FD  11$:    PUSHL   BEG_OF_SIDR                      ; 0596
                                7E  D4 00100          CLRL    -(SP)                            ;
                             0000G  30 00102          BSBW    RM$SQUISH_SIDR                   ;
            5E                  08  C0 00105          ADDL2   #8, SP                           ;
      43    A9        8001      8F  AA 00108          BICW2   #32769, 67(IRAB)                 ; 0597
                                0B  11 0010E          BRB     14$                              ; 0591
                             0000G  30 00110  12$:    BSBW    RM$GETNXT_ARRAY                  ; 0606
                                06  11 00113          BRB     14$                              ; 0591
      04    AE        84EC      8F  3C 00115  13$:    MOVZWL  #34028, ERRSTATUS                ; 0629
    84EC    8F          04      AE  B1 0011B  14$:    CMPW    ERRSTATUS, #34028                ; 0631
                                19  12 00121          BNEQ    17$                              ;
                                61  11 00123          BRB     19$                              ; 0633
      05    A9                  10  88 00125  15$:    BISB2   #16, 5(IRAB)                     ; 0641
                                11  11 00129          BRB     17$                              ; 0521
            54        0084      C9  D0 0012B  16$:    MOVL    132(IRAB), BDB                   ; 0649
                      0084      C9  D4 00130          CLRL    132(IRAB)                        ;
                                7E  D4 00134          CLRL    -(SP)                            ;
                             0000G  30 00136          BSBW    RM$RLSBKT                        ;
            5E                  04  C0 00139          ADDL2   #4, SP                           ;
            54          20      A9  D0 0013C  17$:    MOVL    32(IRAB), BDB                    ; 0651
      0A    A4                  02  88 00140          BISB2   #2, 10(BDB)                      ; 0652
                             0000V  30 00144          BSBW    RM$INS_IF_FIT                    ; 0657
            0B                  50  E9 00147          BLBC    R0, 18$                          ;
                                51  D4 0014A          CLRL    FLAGS                            ; 0668
            54        0084      C9  D0 0014C          MOVL    132(IRAB), BDB                   ; 0670
                                66  12 00151          BNEQ    21$                              ;
                                70  11 00153          BRB     22$                              ; 0677
                             0000G  30 00155  18$:    BSBW    RM$ALLOC_BKT                     ; 0685
      04    AE                  50  D0 00158          MOVL    R0, ERRSTATUS                    ;
            26          04      AE  E9 0015C          BLBC    ERRSTATUS, 19$                   ;
```

```
                                54        3C    A9  D0 00160          MOVL    60(IRAB), BDB                           ; 0690
                                      0000G   30  00164              BSBW    RM$SPLIT_EM                             ; 0694
                             04   AE          50  D0 00167          MOVL    R0, ERRSTATUS
                                  1A    04    AE  E8 0016B          BLBS    ERRSTATUS, 20$
                             0A   A4          01  8A 0016F          BICB2   #1, 10(BDB)                             ; 0697
                                  3C    A9    D4 00173              CLRL    60(IRAB)                                ; 0698
                                        7E    D4 00176              CLRL    -(SP)                                   ; 0699
                                      0000G   30  00178              BSBW    RM$RLSBKT
                                5E            04  C0 0017B          ADDL2   #4, SP
                                50      20    A9  D0 0017E          MOVL    32(IRAB), R0                            ; 0700
                             0A   A0          01  8A 00182          BICB2   #1, 10(R0)
                                      00DF    31  00186 19$:         BRW     29$                                     ; 0696
                           008C   C9    1C    A4  D0 00189 20$:     MOVL    28(BDB), 140(IRAB)                      ; 0706
                             0A   A4          02  88 0018F          BISB2   #2, 10(BDB)                             ; 0707
                                  3C    A9    D4 00193              CLRL    60(IRAB)                                ; 0708
                                      0090    C9  D4 00196          CLRL    144(IRAB)                               ; 0716
                                              02  DD 0019A          PUSHL   #2                                      ; 0720
                                      0000G   30  0019C              BSBW    RM$RLSBKT
                                5E            04  C0 0019F          ADDL2   #4, SP
                             04   AE          50  D0 001A2          MOVL    R0, ERRSTATUS
                                  73    04    AE  E9 001A6          BLBC    ERRSTATUS, 24$
                       21        44    A9        04  E1 001AA      BBC     #4, 68(IRAB), 23$                       ; 0725
                                51            02  D0 001AF          MOVL    #2, FLAGS                               ; 0732
                                54      0084    C9  D0 001B2          MOVL    132(IRAB), BDB                          ; 0734
                                        0C    13  00187              BEQL    22$
                                      0084    C9  D4 001B9 21$:     CLRL    132(IRAB)                               ; 0736
                                        7E    D4 001BD              CLRL    -(SP)                                   ; 0737
                                      0000G   30  001BF              BSBW    RM$RLSBKT
                                5E            04  C0 001C2          ADDL2   #4, SP
                                54      20    A9  D0 001C5 22$:     MOVL    32(IRAB), BDB                           ; 0739
                                20    A9        D4 001C9              CLRL    32(IRAB)                                ; 0740
                                51            DD 001CC              PUSHL   FLAGS                                   ; 0741
                                6A            11 001CE              BRB     27$
                                54      20    A9  D0 001D0 23$:     MOVL    32(IRAB), BDB                           ; 0749
                                55      18    A4  D0 001D4          MOVL    24(BDB), BKT_ADDR                       ; 0750
                           0088   C9    1C    A4  D0 001D8          MOVL    28(BDB), 136(IRAB)                      ; 0751
                       5F     0D   A5          01  E1 001DE          BBC     #1, 13(BKT_ADDR), 28$                   ; 0753
                             15   A7    0C    A5  91 001E3          CMPB    12(BKT_ADDR), 21(IDX_DFN)               ; 0755
                                        58    12 001E8              BNEQ    28$
                             0D   A5          02  8A 001EA          BICB2   #2, 13(BKT_ADDR)                        ; 0762
                                      0000G   30  001EE              BSBW    RM$ALLOC_BRT                            ; 0763
                             04   AE          50  D0 001F1          MOVL    R0, ERRSTATUS
                                  33    04    AE  E9 001F5          BLBC    ERRSTATUS, 25$
                             08   A5    008C  C9  D0 001F9          MOVL    140(IRAB), 8(BKT_ADDR)                  ; 0768
                                54      3C    A9  D0 001FF          MOVL    60(IRAB), BDB                           ; 0772
                                55      18    A4  D0 00203          MOVL    24(BDB), BKT_ADDR                       ; 0773
                                      0000G   30  00207              BSBW    RM$NEW_ROOT                             ; 0774
                             0A   A4          02  88 0020A          BISB2   #2, 10(BDB)                             ; 0778
                                  3C    A9    D4 0020E              CLRL    60(IRAB)                                ; 0779
                                              02  DD 00211          PUSHL   #2                                      ; 0780
                                      0000G   30  00213              BSBW    RM$RLSBKT
                                5E            04  C0 00216          ADDL2   #4, SP
                             04   AE          50  D0 00219          MOVL    R0, ERRSTATUS
                                  0B    04    AE  E9 0021D 24$:     BLBC    ERRSTATUS, 25$
                                      0000G   30  00221              BSBW    RM$UPD_PLG                              ; 0784
                             04   AE          50  D0 00224          MOVL    R0, ERRSTATUS
                                  05    04    AE  E8 00228          BLBS    ERRSTATUS, 26$
```

```
                                 6E                01  D0 0022C 25$:    MOVL    #1, KILL_CUR                            ;
                                                   37  11 0022F        BRB     29$                                     ;
                                 54          20    A9  D0 00231 26$:    MOVL    32(IRAB), BDB                           ; 0788
                                             20    A9  D4 00235         CLRL    32(IRAB)                                ; 0789
                                                   02  DD 00238         PUSHL   #2                                      ; 0790
                                                0000G  30 0023A 27$:    BSBW    RM$RLSBKT                               ;
                                 5E                04  C0 0023D         ADDL2   #4, SP                                  ;
                                                   5B  11 00240         BRB     33$                                     ;
                                                   06  DD 00242 28$:    PUSHL   #6                                      ; 0799
                                                0000G  30 00244         BSBW    RM$RLSBKT                               ;
                                 5E                04  C0 00247         ADDL2   #4, SP                                  ;
                              04 AE                50  D0 0024A         MOVL    R0, ERRSTATUS                           ;
                                 16          04    AE  E9 0024E         BLBC    ERRSTATUS, 29$                          ;
                              08 AE          20    A9  9E 00252         MOVAB   32(R9), 8(SP)                           ; 0800
                              08 BE                54  D0 00257         MOVL    BDB, @8(SP)                             ;
                                             41    A9  96 0025B         INCB    65(IRAB)                                ; 0802
                              42 A9                01  B0 0025E         MOVW    #1, 66(IRAB)                            ; 0803
                                             44    A9  94 00262         CLRB    68(IRAB)                                ; 0804
                                              FDA2  31 00265         BRW     1$                                      ; 0424
                                 54        0084    C9  D0 00268 29$:    MOVL    132(IRAB), BDB                          ; 0810
                                                   0C  13 0026D         BEQL    30$                                     ;
                                           0084    C9  D4 0026F         CLRL    132(IRAB)                               ; 0812
                                                   7E  D4 00273         CLRL    -(SP)                                   ; 0813
                                                0000G  30 00275         BSBW    RM$RLSBKT                               ;
                                 5E                04  C0 00278         ADDL2   #4, SP                                  ;
                                 54          20    A9  D0 0027B 30$:    MOVL    32(IRAB), BDB                           ; 0819
                                                   18  13 0027F         BEQL    32$                                     ;
                                             20    A9  D4 00281         CLRL    32(IRAB)                                ; 0822
                                 0A                6E  E9 00284         BLBC    KILL_CUR, 31$                           ; 0823
                                                   53  D4 00287         CLRL    R3                                      ; 0824
                                     00000000G     EF  16 00289         JSB     RM$RLNERR                               ;
                                                   08  11 0028F         BRB     32$                                     ;
                                                   7E  D4 00291 31$:    CLRL    -(SP)                                   ; 0826
                                                0000G  30 00293         BSBW    RM$RLSBKT                               ;
                              5E                   04  C0 00296         ADDL2   #4, SP                                  ;
                              50          04       AE  D0 00299 32$:    MOVL    ERRSTATUS, R0                           ; 0829
                              5E                   18  C0 0029D 33$:    ADDL2   #24, SP                                 ; 0831
                                                   1C  BA 002A0         POPR    #^M<R2,R3,R4>                           ;
                                                       05 002A2         RSB                                             ;

; Routine Size:  675 bytes,    Routine Base:  RM$RMS3 + 0000
```

```
;  770     0832  1 %SBTTL 'RM$INS_IF_FIT'
;  771     0833  1 GLOBAL ROUTINE RM$INS_IF_FIT : RL$INS_IF_FIT =
;  772     0834  1
;  773     0835  1 !++
;  774     0836  1 !
;  775     0837  1 ! FUNCTIONAL DESCRIPTION:
;  776     0838  1 !
;  777     0839  1 !       This routine inserts a SIDR or index record into the bucket
;  778     0840  1 !       at the position pointed to by REC_ADDR and returns success if
;  779     0841  1 !       it fits else returns 0 to indicate a split is neccessary.
;  780     0842  1 !
;  781     0843  1 ! CALLING SEQUENCE:
;  782     0844  1 !       RM$INS_IF_FIT()
;  783     0845  1 !
;  784     0846  1 ! INPUT PARAMETERS
;  785     0847  1 !       NONE
;  786     0848  1 !
;  787     0849  1 ! IMPLICIT INPUTS:
;  788     0850  1 !       RAB [ LOA ] -           if set use fill sizes to determine bucket size
;  789     0851  1 !       IRAB [ DUPS_SEEN ] - set if duplicates seen meaning only continuation
;  790     0852  1 !                               record is neccessary
;  791     0853  1 !       BKT_ADDR -              points to beginning of bucket
;  792     0854  1 !       IDX_DFN -               pointer to index descriptor
;  793     0855  1 !       [ DATFILL ] -           fill size for data buckets when fill percents used
;  794     0856  1 !       [ IDXFILL ] -           "    "    index    "    "    "       "
;  795     0857  1 !       [ DATBKTSZ ] -          size of data bkts in VBN's
;  796     0858  1 !       [ IDXBKTSZ ] -          "    "    index    "    "
;  797     0859  1 !
;  798     0860  1 ! OUTPUT PARAMETERS:
;  799     0861  1 !       NONE
;  800     0862  1 !
;  801     0863  1 ! IMPLICIT OUTPUTS:
;  802     0864  1 !       NONE
;  803     0865  1 !
;  804     0866  1 ! ROUTINE VALUE:
;  805     0867  1 !       NONE
;  806     0868  1 !
;  807     0869  1 ! SIDE EFFECTS:
;  808     0870  1 !       NONE
;  809     0871  1 !
;  810     0872  1 !--
;  811     0873  1
;  812     0874  2     BEGIN
;  813     0875  2
;  814     0876  2     EXTERNAL REGISTER
;  815     0877  2         R_BKT_ADDR_STR,
;  816     0878  2         R_RAB_STR,
;  817     0879  2         R_IRAB_STR,
;  818     0880  2         R_IFAB_STR,
;  819     0881  2         R_REC_ADDR_STR,
;  820     0882  2         R_IDX_DFN_STR;
;  821     0883  2
;  822     0884  2     GLOBAL REGISTER
;  823     0885  2         R_IMPURE;
;  824     0886  2
;  825     0887  2     LOCAL
;  826     0888  2         REC_SZ;
```

```
;  827          0889  2
;  828          0890  2      ! this block is defined to limit scope of BKT_ROOM
;  829          0891  2      !
;  830          0892  2      BEGIN
;  831          0893  2
;  832          0894  2      LOCAL
;  833          0895  2          END_BKT,
;  834          0896  2          BKT_ROOM        : SIGNED;
;  835          0897  2
;  836          0898  2      ! set up bucket size used to determine split based on whether this is
;  837          0899  2      ! data or index level and whether fill percentages are used
;  838          0900  2      !
;  839          0901  3      IF .BKT_ADDR[BKT$B_LEVEL] EQL 0
;  840          0902  3      THEN
;  841          0903  4          BEGIN
;  842          0904  4          END_BKT = .BKT_ADDR + .IDX_DFN[IDX$B_DATBKTSZ]*512;
;  843          0905  4
;  844          0906  4          IF .RAB[RAB$V_LOA]
;  845          0907  4          THEN
;  846          0908  4              BKT_ROOM = .IDX_DFN[IDX$W_DATFILL]
;  847          0909  4          ELSE
;  848          0910  4              BKT_ROOM = .IDX_DFN[IDX$B_DATBKTSZ]*512;
;  849          0911  4          END
;  850          0912  3      ELSE
;  851          0913  4          BEGIN
;  852          0914  4          END_BKT = .BKT_ADDR + .IDX_DFN[IDX$B_IDXBKTSZ]*512;
;  853          0915  4
;  854          0916  4          IF .RAB[RAB$V_LOA]
;  855          0917  4          THEN
;  856          0918  4              BKT_ROOM = .IDX_DFN[IDX$W_IDXFILL]
;  857          0919  4          ELSE
;  858          0920  4              BKT_ROOM = .IDX_DFN[IDX$B_IDXBKTSZ]*512;
;  859          0921  3          END;
;  860          0922  3
;  861          0923  3      ! Set up record size.
;  862          0924  3      !
;  863          0925  3      REC_SZ = RM$RECORD_SIZE();
;  864          0926  3
;  865          0927  3      ! Establish amount of room left in bucket with new record minus 1 byte for
;  866          0928  3      ! check byte at end of bucket
;  867          0929  3      !
;  868          0930  4      IF  (.IFAB [IFB$B_PLG_VER] GEQU PLG$C_VER_3)
;  869          0931  3           AND
;  870          0932  4          (.BKT_ADDR[BKT$B_LEVEL] GTRU 0)
;  871          0933  3      THEN
;  872          0934  4          BEGIN
;  873          0935  4
;  874          0936  4          LOCAL
;  875          0937  4              VBN_FREE;
;  876          0938  4
;  877          0939  4          VBN_FREE = .END_BKT - BKT$C_ENDOVHD;
;  878          0940  4          BKT_ROOM = .(.VBN_FREE)<0,16> - .BKT_ADDR [BKT$W_FREESPACE];
;  879          0941  4          BKT_ROOM = .BKT_ROOM - .REC_SZ<0,16> - .REC_SZ <16,16>;
;  880          0942  4          END
;  881          0943  3      ELSE
;  882          0944  3          BKT_ROOM = .BKT_ROOM - .REC_SZ - .BKT_ADDR[BKT$W_FREESPACE] - 1;
;  883          0945  3
```

```
;  884          0946  3     IF .BKT_ROOM LSS 0
;  885          0947  3     THEN
;  886          0948  3         RETURN 0;
;  887          0949  3
;  888          0950  3     IRAB[IRB$W_POS_INS] = .REC_ADDR - .BKT_ADDR; ! set up for INS_REC
;  889          0951  2     END; ! of block defining BKT_ROOM
;  890          0952  2     RETURN RM$INS_REC(.BKT_ADDR, .REC_SZ);
;  891          0953  2
;  892          0954  1     END;


                                    080C   8F  BB 00000 RM$INS_IF_FIT::
                                                        PUSHR   #^M<R2,R3,R11>                          ; 0833
                                      0C    A5  95 00004        TSTB    12(BKT_ADDR)                      ; 0901
                                            17  12 00007        BNEQ    1$                                ;
                                      17    A7  9A 00009        MOVZBL  23(IDX_DFN), R0                   ; 0904
                                            09  78 0000D        ASHL    #9, R0, R0                        ;
                                            50  C1 00011        ADDL3   R0, BKT_ADDR, END_BKT             ;
                                      1D    A8  05 E1 00015     BBC     #5, 5(RAB), 2$                    ; 0906
                                      26    A7  3C 0001A        MOVZWL  38(IDX_DFN), BKT_ROOM             ; 0908
                                            1A  11 0001E        BRB     3$                                ;
                                      16    A7  9A 00020 1$:    MOVZBL  22(IDX_DFN), R0                   ; 0914
                                            09  78 00024        ASHL    #9, R0, R0                        ;
                                            50  C1 00028        ADDL3   R0, BKT_ADDR, END_BKT             ;
                                      06    A8  05 E1 0002C     BBC     #5, 5(RAB), 2$                    ; 0916
                                      24    A7  3C 00031        MOVZWL  36(IDX_DFN), BKT_ROOM             ; 0918
                                            03  11 00035        BRB     3$                                ;
                                            50  D0 00037 2$:    MOVL    R0, BKT_ROOM                      ; 0920
                                    0000G   30  00 0003A 3$:    BSBW    RM$RECORD_SIZE                    ; 0925
                                    00B7    CA  91 0003D        CMPB    183(IFAB), #3                     ; 0930
                                            25  1F 00042        BLSSU   4$                                ;
                                      0C    A5  95 00044        TSTB    12(BKT_ADDR)                      ; 0932
                                            20  13 00047        BEQL    4$                                ;
                                      FC    A3  9E 00049        MOVAB   -4(R3), VBN_FREE                  ; 0939
                                            61  3C 0004D        MOVZWL  (VBN_FREE), BKT_ROOM              ; 0940
                                      04    A5  3C 00050        MOVZWL  4(BKT_ADDR), R3                   ;
                                            53  C2 00054        SUBL2   R3, BKT_ROOM                      ;
                                            50  3C 00057        MOVZWL  REC_SZ, R1                        ; 0941
                                            51  C3 0005A        SUBL3   R1, BKT_ROOM, R1                  ;
                                        10  10  EF 0005E        EXTZV   #16, #16, REC_SZ, BKT_ROOM        ;
                                            51  C3 00063        SUBL3   BKT_ROOM, R1, BKT_ROOM            ;
                                            0F  11 00067        BRB     5$                                ; 0930
                                            50  C3 00069 4$:    SUBL3   REC_SZ, BKT_ROOM, R1              ; 0944
                                      04    A5  3C 0006D        MOVZWL  4(BKT_ADDR), R3                   ;
                                            53  C2 00071        SUBL2   R3, R1                            ;
                                      FF    A1  9E 00074        MOVAB   -1(R1), BKT_ROOM                  ;
                                            11  19 00078 5$:    BLSS    6$                                ; 0946
                                  48    A9  56  C3 0007A        SUBW3   BKT_ADDR, REC_ADDR, 72(IRAB)      ; 0950
                                            50  DD 0007F        PUSHL   REC_SZ                            ; 0952
                                            55  DD 00081        PUSHL   BKT_ADDR                          ;
                                    0000G   30  00 00083        BSBW    RM$INS_REC                        ;
                                        5E  CO  08 00086        ADDL2   #8, SP                            ;
                                            02  11 00089        BRB     7$                                ;
                                            50  D4 0008B 6$:    CLRL    R0                                ; 0954
```

```
                                080C   8F  BA 0008D 7$:     POPR    #^M<R2,R3,R11>                  ;
                                           05 00091         RSB                                     ;

; Routine Size:  146 bytes,    Routine Base:  RM$RMS3 + 02A3

;  893          0955  1
;  894          0956  1 END
;  895          0957  1
;  896          0958  0 ELUDOM



;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  RM$RMS3                       821  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        80         2         154        00:00.4



;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3UPSIDX/OBJ=OBJ$:RM3UPSIDX MSRC$:RM3UPSIDX/UPDATE=(ENH$:RM3UPSIDX)

; Size:          821 code + 0 data bytes
; Run Time:          00:21.6
; Elapsed Time:      01:00.1
; Lines/CPU Min:      2656
; Lexemes/CPU-Min:  15848
; Memory Used:  263 pages
; Compilation Complete
```

RM3SSIDR
LIS

RM3UPSIDX
LIS

RM3SRCHKY
LIS

RM3UPDDEL
LIS

RM3UPDATE
LIS

RMS0FSCN
LIS

RM3XALLO
LIS

RMS0CONN
LIS

RMS0ENTER
LIS

RMS0ERASE
LIS

RMS0CREAT
LIS

RMS0DELET
LIS

RMS0FIND
LIS

RM3XSUMO
LIS

RMS0DISPL
LIS

RM3XKEYO
LIS

RMS0GET
LIS

RMS0CLOSE
LIS

RMS0DISC
LIS

RMS0EXTEN
LIS

RMS0BLKIO
LIS