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

```
RRRRRRRR    MM      MM    333333      SSSSSSSS  RRRRRRRR      CCCCCCCC  HH      HH  KK      KK  YY      YY
RRRRRRRR    MM      MM    333333      SSSSSSSS  RRRRRRRR      CCCCCCCC  HH      HH  KK      KK  YY      YY
RR      RR  MMMM  MMMM  33      33  SS          RR      RR  CC          HH      HH  KK      KK  YY      YY
RR      RR  MMMM  MMMM  33      33  SS          RR      RR  CC          HH      HH  KK      KK  YY      YY
RR      RR  MM  MM  MM        33  SS          RR      RR  CC          HH      HH  KK    KK      YY  YY
RR      RR  MM  MM  MM        33  SS          RR      RR  CC          HH      HH  KK    KK      YY  YY
RRRRRRRR    MM      MM      33      SSSSSS    RRRRRRRR    CC          HHHHHHHHHH  KKKKKK          YY
RRRRRRRR    MM      MM      33      SSSSSS    RRRRRRRR    CC          HHHHHHHHHH  KKKKKK          YY
RR  RR      MM      MM        33          SS  RR  RR      CC          HH      HH  KK    KK        YY
RR  RR      MM      MM        33          SS  RR  RR      CC          HH      HH  KK    KK        YY
RR    RR    MM      MM  33    33          SS  RR    RR    CC          HH      HH  KK      KK      YY
RR    RR    MM      MM  33    33          SS  RR    RR    CC          HH      HH  KK      KK      YY
RR      RR  MM      MM    333333    SSSSSSSS  RR      RR    CCCCCCCC  HH      HH  KK      KK      YY
RR      RR  MM      MM    333333    SSSSSSSS  RR      RR    CCCCCCCC  HH      HH  KK      KK      YY

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

```
;     1          0001  0 MODULE RM3SRCHKY (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 !                This module starts at current bucket and searchs to stop level
;    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:        D. H. Gillespie     CREATION DATE:             18-APR-78  9:39
;    45          0045  1 !
;    46          0046  1 ! MODIFIED BY:
;    47          0047  1 !
;    48          0048  1 !       V03-011 TMK0004         Todd M. Katz            03-Apr-1983
;    49          0049  1 !               Add support for RU record space reclamation. This involves
;    50          0050  1 !               modifications to RM$SEARCH_TREE as follows. Whenever RMS
;    51          0051  1 !               has write access to a file that is marked RU Journallable,
;    52          0052  1 !               and is currently positioning to a primary data bucket but
;    53          0053  1 !               not for modification, then RMS wants to set the CSH$V_LOCK
;    54          0054  1 !               flag so that the bucket will be EXclusively locked in case RU
;    55          0055  1 !               record reclamation takes place on the bucket. In addition, RMS
;    56          0056  1 !               wants to EXclusively lock all SIDR buckets whenever it has
;    57          0057  1 !               write access to the file.
```

```
;    58          0058  1 !
;    59          0059  1 !       V03-010 MCN0005         Maria del C. Nasr       15-Mar-1983
;    60          0060  1 !               More linkages reorganization
;    61          0061  1 !
;    62          0062  1 !       V03-009 MCN0004         Maria del C. Nasr       28-Feb-1983
;    63          0063  1 !               Reorganize linkages
;    64          0064  1 !
;    65          0065  1 !       V03-008 TMK0003         Todd M. Katz            03-Feb-1983
;    66          0066  1 !               Add support for Recovery Unit Journalling and RU ROLLBACK
;    67          0067  1 !               Recovery of ISAM files. Make a change to SEARCH_V2. Do not set
;    68          0068  1 !               the bit IRB$V_DUPS_SEEN if the current primary data record is
;    69          0069  1 !               either marked RU_DELETE or DELETED. Previously just a check for
;    70          0070  1 !               DELETED was being made.
;    71          0071  1 !
;    72          0072  1 !       V03-007 TMK0002         Todd M. Katz            09-Sep-1982
;    73          0073  1 !               The field IRB$B_SRCHFLAGS has been changed to a word in size.
;    74          0074  1 !               Fix all references to it.
;    75          0075  1 !
;    76          0076  1 !               Add support for prologue 3 SIDRs. This includes the removal of
;    77          0077  1 !               misleading comment statements.
;    78          0078  1 !
;    79          0079  1 !               In the local routine SEARCH_V2, when a search key is found to
;    80          0080  1 !               be equal to a key of a record, set the IRAB bit IRB$V_DUP_KEY.
;    81          0081  1 !
;    82          0082  1 !
;    83          0083  1 !       V03-006 KBT0297         Keith B. Thompson       24-Aug-1982
;    84          0084  1 !               Reorganize psects
;    85          0085  1 !
;    86          0086  1 !       V03-005 MCN0003         Maria del C. Nasr       29-Jun-1982
;    87          0087  1 !               Allow keys of different data types other than string
;    88          0088  1 !               in prologue 3 files.  Change all CH$COMPARE calls to
;    89          0089  1 !               RM$COMPARE_KEY to compare keys taking into consideration
;    90          0090  1 !               the different data types.
;    91          0091  1 !
;    92          0092  1 !       V03-004 TMK0001         Todd M. Katz            25-May-1982
;    93          0093  1 !               Eliminate unneccesary linkage declaration for RM$RECORD_KEY.
;    94          0094  1 !
;    95          0095  1 !       V03-003 MCN0002         Maria del C. Nasr       25-Mar-1982
;    96          0096  1 !               Use macro to calculate key buffer address.
;    97          0097  1 !
;    98          0098  1 !       V03-002 CDS0008         C Saether               23-Mar-1982
;    99          0099  1 !               Restore BKT_ADDR when BDB is restored when leaving
;   100          0100  1 !               block that confirms down pointer from level 1.
;   101          0101  1 !               This was causing the check for level 0 to fail and
;   102          0102  1 !               not lock the next bucket when positioning horizontally
;   103          0103  1 !               at level 0.
;   104          0104  1 !
;   105          0105  1 !       V03-001 CDS0007         C Saether                3-Mar-1982
;   106          0106  1 !               Fix bug where horizontal reads at the stoplevel were
;   107          0107  1 !               not locking the bucket on positions for insert or
;   108          0108  1 !               delete.  This was working correctly for position for
;   109          0109  1 !               insert at level 0 only (which is the normal place
;   110          0110  1 !               where horizontal reads would take place).
;   111          0111  1 !               The changes in RM$CACHE and RM$RELEASE have made them
;   112          0112  1 !               sensitive to this error.
;   113          0113  1 !
;   114          0114  1 !       V02-039 CDS0006         C Saether               20-Jan-1982                ;
```

```
;  115          0115  1 !                Set VBN_LEFT for scan forward when not locking.
;  116          0116  1 !                Clear IRB$L_REC_COUNT before calling RM$SRCH_BY_KEY
;  117          0117  1 !                in the horizontal scan logic.
;  118          0118  1 !                Once data level horizontal scan has started, do not
;  119          0119  1 !                drop back into outer loop anymore.
;  120          0120  1 !
;  121          0121  1 !        V02-038 CDS0005         C Saether               14-Jan-1982
;  122          0122  1 !                Put back logic to check index down pointer after
;  123          0123  1 !                positioning to level on insert.  It's really necessary.
;  124          0124  1 !
;  125          0125  1 !        V02-037 CDS0004         C Saether               31-Dec-1981
;  126          0126  1 !                Fix error path botched in V02-036.
;  127          0127  1 !                Put back retry on rlk error going down tree inadvertently
;  128          0128  1 !                removed by V02-036.
;  129          0129  1 !
;  130          0130  1 !        V02-036 CDS0003         C Saether               29-Dec-1981
;  131          0131  1 !                Remove logic to check down pointer before going
;  132          0132  1 !                across on position for insert.
;  133          0133  1 !                Modify horizontal scan at data level on position for
;  134          0134  1 !                insert to release bucket prior to going ahead.  This
;  135          0135  1 !                eliminates need for special algorithms to avoid deadlock
;  136          0136  1 !                with rrv updating by other streams.  Previous algorithm
;  137          0137  1 !                could get into infinite loop with another process.
;  138          0138  1 !
;  139          0139  1 !        V02-035 PSK0009         P Knibbe                07-Oct-1981
;  140          0140  1 !                Make sure all calls to RM$RECORD_VBN check for
;  141          0141  1 !                prologue version.
;  142          0142  1 !
;  143          0143  1 !        V02-034 CDS0002         C Saether               29-Sep-1981
;  144          0144  1 !                Make check for exclusive lock before setting abovelckd.
;  145          0145  1 !
;  146          0146  1 !        V02-033 CDS0001         C Saether               20-Aug-1981
;  147          0147  1 !                Remove references to BDB$L_OWN.
;  148          0148  1 !
;  149          0149  1 !        V02-032 PSK0008         Paulina S. Knibbe       08-Aug-1981
;  150          0150  1 !                Remove all references to SPLCTX
;  151          0151  1 !
;  152          0152  1 !        V02-031 PSK0007         Paulina S. Knibbe       30-Jul-1981
;  153          0153  1 !                Remove support for truncated index keys.
;  154          0154  1 !
;  155          0155  1 !        V02-030 PSK0006         Paulina S. Knibbe       12-Jun-1981
;  156          0156  1 !                1) Make RM$V3_VBN a global routine
;  157          0157  1 !                2) Save LST_NCMP when bouncing back and forth between
;  158          0158  1 !                   two buckets.
;  159          0159  1 !                3) Add logic to support horizontal processing in V3
;  160          0160  1 !                   compresses indexes.
;  161          0161  1 !                4) Add RM$CHK_HORIZ
;  162          0162  1 !                5) Change calls to COMP_ASCII to be CH$COMPARE
;  163          0163  1 !
;  164          0164  1 !        V02-029 PSK0005         Paulina S. Knibbe       28-May-1981
;  165          0165  1 !                Fix problem on GT search. Make lock-above optimization
;  166          0166  1 !                work for prologue three files.
;  167          0167  1 !
;  168          0168  1 !        V02-028 PSK0004         Paulina S. Knibbe       03-May-1981
;  169          0169  1 !                Add support for compressed keys in index, SIDR and data
;  170          0170  1 !                buckets.
;  171          0171  1 !
```

```
;  172          0172  1 !       V02-027 PSK0003         Paulina S. Knibbe       30-Apr-1981
;  173          0173  1 !               Fix problem when key goes past range in bucket
;  174          0174  1 !
;  175          0175  1 !       V02-026 MCN0001         Maria del C. Nasr       23-Apr-1981
;  176          0176  1 !               Add support for non-compressed keys at the primary data
;  177          0177  1 !               level of prologue 3 files.
;  178          0178  1 !
;  179          0179  1 !       V02-025 PSK0002         Paulina S. Knibbe       30-Mar-1981
;  180          0180  1 !               Add support for random searching of PLG3 buckets with
;  181          0181  1 !               fixed length keys.
;  182          0182  1 !
;  183          0183  1 !       V02-024 PSK0001         Paulina S. Knibbe       19-Mar-1981
;  184          0184  1 !               Change index bucket VBN processing to handle VBN lists
;  185          0185  1 !               at end of bucket
;  186          0186  1 !
;  187          0187  1 !       V02-023 REFORMAT        Keith B. Thompson       23-Jul-1980
;  188          0188  1 !
;  189          0189  1 ! REVISION HISTORY:
;  190          0190  1 !
;  191          0191  1 !       Christian Saether,      27-JUL-78  10:48
;  192          0192  1 !       X0002 - SRCH_BY_KEY changed to store last record address for
;  193          0193  1 !               ADD_TO_ARRAY
;  194          0194  1 !
;  195          0195  1 !       Christian Saether,      9-AUG-78  11:55
;  196          0196  1 !       X0003 - never return RNF on POSINSERT
;  197          0197  1 !
;  198          0198  1 !       Wendy Koenig,   11-AUG-78  14:10
;  199          0199  1 !       X0004 - only lock the root if it is the lock-above or stoplevel level
;  200          0200  1 !
;  201          0201  1 !       Christian Saether,      14-AUG-78  10:21
;  202          0202  1 !       X0005 - add code to disable RNF on position for insert
;  203          0203  1 !
;  204          0204  1 !       Christian Saether,      24-AUG-78  9:39
;  205          0205  1 !       X0006 - Add POSDELETE to SEARCH_TREE
;  206          0206  1 !
;  207          0207  1 !       Christian Saether,      24-AUG-78  19:41
;  208          0208  1 !       X0007 - Correct logic error horizontal positioning on insert across
;  209          0209  1 !               empty buckets
;  210          0210  1 !
;  211          0211  1 !       Christian Saether,      26-SEP-78  15:57
;  212          0212  1 !       X0008 - never do lock_above on position for delete
;  213          0213  1 !
;  214          0214  1 !       D. H. Gillespie,        6-OCT-78  17:34
;  215          0215  1 !       X0009 - fix bug when positioning for insert and entire file is empty
;  216          0216  1 !               from where search started and end
;  217          0217  1 !
;  218          0218  1 !       Christian Saether,      9-OCT-78  12:10
;  219          0219  1 !       X0010 - reverse usage of EMPT_SEEN and EMPTY_BKT so EMPTY_BKT is not
;  220          0220  1 !               clobbered while searching index levels
;  221          0221  1 !
;  222          0222  1 !       Wendy Koenig,   24-OCT-78  14:03
;  223          0223  1 !       X0011 - make changes caused by sharing conventions
;  224          0224  1 !
;  225          0225  1 !       Christian Saether,      27-OCT-78  15:09
;  226          0226  1 !       X0012 - do not set DUPS_SEEN if the record is deleted on position for
;  227          0227  1 !               insert
;  228          0228  1 !
```

```
;  229     0229  1 !       Christian Saether,      12-DEC-78  20:17
;  230     0230  1 !       X0013 - set ADUP_SEEN always when passing dupes - this has been taken
;  231     0231  1 !               out 12/18/78
;  232     0232  1 !
;  233     0233  1 !       Christian Saether,      15-DEC-78  11:17
;  234     0234  1 !       X0014 - mask split_bits in csearch_tree before calling search_tree
;  235     0235  1 !
;  236     0236  1 !       Christian Saether,      9-JAN-79  12:08
;  237     0237  1 !       X0015 - on lock error going horizontal, reread bucket with lock to
;  238     0238  1 !               force stall
;  239     0239  1 !
;  240     0240  1 !       Christian Saether,      15-JAN-79  18:22
;  241     0241  1 !       X0016 - modify lockabove logic to avoid deadlocks
;  242     0242  1 !
;  243     0243  1 !       Christian Saether,      20-JAN-79  20:18
;  244     0244  1 !       X0017 - confirm level 1 down pointer before going across level 0
;  245     0245  1 !
;  246     0246  1 !       Wendy Koenig,   23-JAN-79  14:21
;  247     0247  1 !       X0018 - zero MIDX_TMP1
;  248     0248  1 !
;  249     0249  1 !       Christian Saether,      24-JAN-79  11:25
;  250     0250  1 !       X0019 - set up bkt_addr after releasing empty bucket when going across
;  251     0251  1 !               at data level
;  252     0252  1 !
;  253     0253  1 !       Christian Saether,      26-JAN-79  9:49
;  254     0254  1 !       X0020 - only clear splits bits when coming around from data level,
;  255     0255  1 !               take out going across at data level kluge
;  256     0256  1 !
;  257     0257  1 !       Christian Saether,      6-FEB-79  17:42
;  258     0258  1 !       X0021 - do not look at owner of bdb to make lockabove decision
;  259     0259  1 !
;  260     0260  1 !       Christian Saether,      14-FEB-79  14:05
;  261     0261  1 !       X0022 - always go across data level primary key dupes
;  262     0262  1 !
;  263     0263  1 !*****
;  264     0264  1
;  265     0265  1 LIBRARY 'RMSLIB:RMS';
;  266     0266  1
;  267     0267  1 REQUIRE 'RMSSRC:RMSIDXDEF';
;  268     0332  1
;  269     0333  1 ! define default psects for code
;  270     0334  1 !
;  271     0335  1 PSECT
;  272     0336  1     CODE = RM$RMS3(PSECT_ATTR),
;  273     0337  1     PLIT = RM$RMS3(PSECT_ATTR);
;  274     0338  1
;  275     0339  1 !
;  276     0340  1
;  277     0341  1 MACRO
;  278   M 0342  1     L_HIGH_KEY =
;  279     0343  1         RL$HIGH_KEY = JSB()%;
;  280     0344  1
;  281     0345  1
;  282     0346  1 ! Linkages
;  283     0347  1 !
;  284     0348  1
;  285     0349  1 LINKAGE
```

```
;  286          0350  1      L_CACHE,
;  287          0351  1      L_COMPARE_KEY,
;  288          0352  1      L_HIGH_KEY,
;  289          0353  1      L_PRESERVE1,
;  290          0354  1      L_RABREG_4567,
;  291          0355  1      L_RABREG_457,
;  292          0356  1      L_RABREG_567,
;  293          0357  1      L_RABREG_67,
;  294          0358  1      L_RABREG_7,
;  295          0359  1      L_REC_OVHD;
;  296          0360  1
;  297          0361  1 !
;  298          0362  1 ! Forward Routines
;  299          0363  1 !
;  300          0364  1
;  301          0365  1 FORWARD ROUTINE
;  302          0366  1      RM$V3_VBN           : RL$RABREG_567;
;  303          0367  1
;  304          0368  1 !
;  305          0369  1 ! External Routines
;  306          0370  1 !
;  307          0371  1
;  308          0372  1 EXTERNAL ROUTINE
;  309          0373  1      RM$CACHE            : RL$CACHE ADDRESSING_MODE( GENERAL ),
;  310          0374  1      RM$CNTRL_ADDR       : RL$RABREG_567,
;  311          0375  1      RM$COMPARE_KEY      : RL$COMPARE_KEY,
;  312          0376  1      RM$GETBKT           : RL$RABREG_457,
;  313          0377  1      RM$KEY_DESC         : RL$RABREG_7,
;  314          0378  1      RM$REC_OVHD         : RL$REC_OVHD,
;  315          0379  1      RM$RECORD_VBN       : RL$PRESERVE1,
;  316          0380  1      RM$RLSBKT           : RL$PRESERVE1,
;  317          0381  1      RM$SRCH_CMPR        : RL$RABREG_567;
;  318          0382  1
;  319          0383  1 LITERAL
;  320          0384  1      LS = -1,
;  321          0385  1      GT = 1,
;  322          0386  1      EQ = 0;
```

```
;  324     0387  1 ROUTINE SEARCH_FIX (GOAL) : RL$RABREG_567 =
;  325     0388  1
;  326     0389  1 !++
;  327     0390  1 !
;  328     0391  1 ! SEARCH_FIX
;  329     0392  1 !
;  330     0393  1 !       This routine searches a PLG3 fixed length key index bucket from
;  331     0394  1 !       the current record address to the end of the bucket for an index
;  332     0395  1 !       record equal or greater than the input search key.
;  333     0396  1 !
;  334     0397  1 ! CALLING SEQUENCE:
;  335     0398  1 !       SEARCH_FIX
;  336     0399  1 !
;  337     0400  1 ! INPUT PARAMETERS:
;  338     0401  1 !
;  339     0402  1 !       GOAL - 0 - be satisfied with an equal match
;  340     0403  1 !            - 1 - position past an equal match (search for GT match)
;  341     0404  1 !
;  342     0405  1 ! IMPLICIT INPUTS:
;  343     0406  1 !       REC_ADDR                - address of record in bucket to begin search on
;  344     0407  1 !       BKT_ADDR                - address of current bucket
;  345     0408  1 !       IDX_DFN                 - address of index descriptor for current key of reference
;  346     0409  1 !       IRAB                    - address of internal RAB
;  347     0410  1 !       IRAB[IRB$V_SRCHGT]      - if set, search for index record gt search key
;  348     0411  1 !       IRAB[IRB$V_POSINSERT]   - if set, search for position to insert record
;  349     0412  1 !       IRAB[KEY BUFFER 2]      - address of search key
;  350     0413  1 !       IRAB[IRB$B_KEYSZ]       - size of key to compare
;  351     0414  1 !
;  352     0415  1 ! OUTPUT PARAMETERS:
;  353     0416  1 !       NONE
;  354     0417  1 !
;  355     0418  1 ! IMPLICIT OUTPUTS:
;  356     0419  1 !       REC_ADDR        - if EQ, address of index record equal to search key
;  357     0420  1 !                       - if GT, at end of record data in bucket
;  358     0421  1 !                       - if LS, address of index record greater than search key
;  359     0422  1 !       IRAB [IRB$L_REC_COUNT]  - Numbe- of the record who address is in REC_ADDR
;  360     0423  1 !       [DUPS_SEEN]     - Set if goal is gt and we found an eq match
;  361     0424  1 !
;  362     0425  1 ! ROUTINE VALUE:
;  363     0426  1 !       R0              -  0, search key = index record
;  364     0427  1 !                       - -1, search key < index record
;  365     0428  1 !                       - +1, search key > index record
;  366     0429  1 !
;  367     0430  1 ! SIDE EFFECTS:
;  368     0431  1 !
;  369     0432  1 !--
;  370     0433  2 BEGIN
;  371     0434  2
;  372     0435  2 EXTERNAL REGISTER
;  373     0436  2     R_IRAB_STR,
;  374     0437  2     R_IFAB_STR,
;  375     0438  2     R_IDX_DFN_STR,
;  376     0439  2     R_BKT_ADDR_STR,
;  377     0440  2     R_REC_ADDR_STR;
;  378     0441  2
;  379     0442  2 LOCAL
;  380     0443  2     STATUS,             ! place to hold results of compare
```

```
;  381          0444  2     RSTART,                 ! address of first record in range
;  382          0445  2     REND,                   ! address of last record in range
;  383          0446  2     EOB;                    ! address past last record in bucket
;  384          0447  2
;  385          0448  2 EOB = .BKT_ADDR + .BKT_ADDR [BKT$W_FREESPACE];
;  386          0449  2 RSTART = .REC_ADDR;
;  387          0450  2 REND = .EOB;
;  388          0451  2
;  389          0452  2 ! Do a binary search of fixed length index records
;  390          0453  2 !
;  391          0454  2
;  392          0455  2 WHILE 1 DO
;  393          0456  3     BEGIN
;  394          0457  3     LOCAL
;  395          0458  3         SIZE;               ! Number of characters in range
;  396          0459  3
;  397          0460  3     BUILTIN
;  398          0461  3         AP;
;  399          0462  3
;  400          0463  3     SIZE = .REND - .RSTART;
;  401          0464  3     REC_ADDR = (.RSTART + (.SIZE^-1) / .IDX_DFN[IDX$B_KEYSZ] * .IDX_DFN[IDX$B_KEYSZ]);
;  402          0465  3     AP = 3;                 ! Contiguous compare
;  403          0466  3     STATUS = RM$COMPARE_KEY (.REC_ADDR, KEYBUF_ADDR(2), .IRAB [IRB$B_KEYSZ]);
;  404          0467  3
;  405          0468  3     CASE .STATUS FROM LS TO GT OF
;  406          0469  3         SET
;  407          0470  3         [LS]:   ! Search key is < record in bucket
;  408          0471  3                 ! prepare to search low half of bucket
;  409          0472  3                 !
;  410          0473  4                 BEGIN
;  411          0474  4                 REND = .REC_ADDR;
;  412          0475  4
;  413          0476  4                 IF .REND EQL .RSTART
;  414          0477  4                 THEN
;  415          0478  4
;  416          0479  4                     ! Only one record left and we just compared it
;  417          0480  4                     !
;  418          0481  4                     EXITLOOP;
;  419          0482  3                 END;
;  420          0483  3
;  421          0484  3         [EQ]:   ! Search key = record in bucket
;  422          0485  3                 !
;  423          0486  3                 !       GENERIC SEARCH          GOAL IS EQ MATCH
;  424          0487  3                 !           yes                     yes             search previous
;  425          0488  3                 !           yes                     no              search next
;  426          0489  3                 !           no                      yes             return
;  427          0490  3                 !           no                      no              search next
;  428          0491  3                 !
;  429          0492  4                 BEGIN
;  430          0493  4
;  431          0494  4                 IF .GOAL EQL EQ
;  432          0495  4                 THEN
;  433          0496  4                     IF .IRAB [IRB$B_KEYSZ] EQL .IDX_DFN [IDX$B_KEYSZ]
;  434          0497  4                     THEN
;  435          0498  4                         EXITLOOP
;  436          0499  4                     ELSE
;  437          0500  4
```

```
;  438     0501  4                    ! generic equal match, we must postion to the previous
;  439     0502  4                    ! record in order to find first equal match
;  440     0503  4                    !
;  441     0504  5                    BEGIN
;  442     0505  5
;  443     0506  5                    IF (.REND - .RSTART) EQL .IDX_DFN [IDX$B_KEYSZ]
;  444     0507  5                    THEN
;  445     0508  5                        EXITLOOP;
;  446     0509  5
;  447     0510  6                    IF (.REND - .RSTART) EQL (.IDX_DFN [IDX$B_KEYSZ]^1)
;  448     0511  5                    THEN
;  449     0512  5                        REND = .REC_ADDR
;  450     0513  5                    ELSE
;  451     0514  5                        REND = .REC_ADDR + .IDX_DFN [IDX$B_KEYSZ];
;  452     0515  5                    END
;  453     0516  5
;  454     0517  4                ELSE
;  455     0518  4                    !
;  456     0519  4                    ! Search past the equal match
;  457     0520  4                    !
;  458     0521  5                    BEGIN
;  459     0522  5                    !
;  460     0523  5                    ! If there is another record in the bucket, compare it
;  461     0524  5                    !
;  462     0525  5                    REC_ADDR = .REC_ADDR + .IDX_DFN[IDX$B_KEYSZ];
;  463     0526  5
;  464     0527  5                    IF .REC_ADDR EQL .EOB
;  465     0528  5                    THEN
;  466     0529  5                        STATUS = GT
;  467     0530  5                    ELSE
;  468     0531  5                        STATUS = LS;
;  469     0532  5                    !
;  470     0533  5                    EXITLOOP;
;  471     0534  5                    END
;  472     0535  3                END;
;  473     0536  3                !
;  474     0537  3            [GT]:   ! Search key is > record in bucket
;  475     0538  3                    !
;  476     0539  4                    BEGIN
;  477     0540  4                    REC_ADDR = .REC_ADDR + .IDX_DFN[IDX$B_KEYSZ];
;  478     0541  4                    !
;  479     0542  4                    IF (.REND - .RSTART) LEQ (.IDX_DFN[IDX$B_KEYSZ])^1
;  480     0543  4                    THEN
;  481     0544  4                    !
;  482     0545  4                        ! We just compared the last record in the range
;  483     0546  4                        ! If this is the last record in the bucket, return
;  484     0547  4                        ! with a GT status.
;  485     0548  4                        !
;  486     0549  4                    !
;  487     0550  4                        IF .REND EQL .EOB
;  488     0551  4                        THEN
;  489     0552  4                            EXITLOOP
;  490     0553  4                        ELSE
;  491     0554  4                            ! We are not at the end of bucket so we know that
;  492     0555  4                            ! we must have done an earlier compare against the
;  493     0556  4                            ! low record of the next range and got an LS result.
;  494     0557  4                            ! Return that record.
```

```
;  495          0558  4                          !
;  496          0559  5                          BEGIN
;  497          0560  5                          STATUS = LS;
;  498          0561  5                          EXITLOOP;
;  499          0562  4                          END;
;  500          0563  4
;  501          0564  4                      ! Prepare to search high half of bucket
;  502          0565  4                      !
;  503          0566  4                      RSTART = .REC_ADDR;
;  504          0567  3                      END;
;  505          0568  3                  TES;
;  506          0569  2              END;
;  507          0570  2
;  508          0571  2  ! We now have a record of some sort.
;  509          0572  2  ! Compute the record count so that we can later pick up the VBN portion
;  510          0573  2  ! of the record from the end of the bucket
;  511          0574  2  !
;  512          0575  2  IRAB[IRB$L_REC_COUNT] = (.REC_ADDR-.BKT_ADDR-BKT$C_OVERHDSZ)/.IDX_DFN[IDX$B_KEYSZ];
;  513          0576  2
;  514          0577  2  RETURN .STATUS;
;  515          0578  1  END;


                                                .TITLE  RM3SRCHKY
                                                .IDENT  \V04-000\

                                                .EXTRN  RM$CACHE, RM$CNTRL_ADDR
                                                .EXTRN  RM$COMPARE_KEY, RM$GETBKT
                                                .EXTRN  RM$KEY_DESC, RM$REC_OVHD
                                                .EXTRN  RM$RECORD_VBN, RM$RMSBKT
                                                .EXTRN  RM$SRCH_CMPR

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

                        091C    8F   BB 00000 SEARCH_FIX:
                                                PUSHR   #^M<R2,R3,R4,R8,R11>                    ; 0387
                     50         04   A5  3C 00004          MOVZWL  4(BKT_ADDR), R0                 ; 0448
                                6045 9F  00008          PUSHAB  (R0)[BKT_ADDR]
                     58              56  D0 0000B          MOVL    REC_ADDR, RSTART                ; 0449
                     52              6E  D0 0000E          MOVL    EOB, REND                       ; 0450
               54    52              58  C3 00011 1$:      SUBL3   RSTART, REND, R4                ; 0463
                     50              54  D0 00015          MOVL    R4, SIZE
               50    50         FF   8F  78 00018          ASHL    #-1, SIZE, R0                   ; 0464
                     51         20   A7  9A 0001D          MOVZBL  32(IDX_DFN), R1
                     50              51  C6 00021          DIVL2   R1, R0
                     53         20   A7  9A 00024          MOVZBL  32(IDX_DFN), R3
                     50              53  C4 00028          MULL2   R3, R0
               56    50              58  C1 0002B          ADDL3   RSTART, R0, REC_ADDR
                     5C              03  D0 0002F          MOVL    #3, AP                          ; 0465
                     53         00B4 CA  3C 00032          MOVZWL  180(IFAB), R3                   ; 0466
                     53         60   A9  C0 00037          ADDL2   96(IRAB), R3
                     50         00A6 C9  9A 0003B          MOVZBL  166(IRAB), R0
                     51              56  D0 00040          MOVL    REC_ADDR, R1
                                0000G 30 00043          BSBW    RM$COMPARE_KEY
                     5B              50  D0 00046          MOVL    R0, STATUS
                  02 FFFFFFFF   8F   5B  CF 00049          CASEL   STATUS, #-1, #2                 ; 0468
            0051          0010        0006      00051 2$:      .WORD   3$-2$,-
```

```
                                                                                    4$-2$,-
                                                                                    8$-2$
                52              56  D0 00057 3$:    MOVL    REC_ADDR, REND          ; 0474
                58              52  D1 0005A        CMPL    REND, RSTART            ; 0476
                                B2  12 0005D        BNEQ    1$
                                64  11 0005F        BRB     11$                     ; 0481
                          1C    AE  D5 00061 4$:    TSTL    GOAL                    ; 0494
                                2B  12 00064        BNEQ    7$
                    20  A7  00A6  C9  91 00066      CMPB    166(IRAB), 32(IDX_DFN)  ; 0496
                                57  13 0006C        BEQL    11$
          54        20  A7        08  00 ED 0006E  CMPZV   #0, #8, 32(IDX_DFN), R4  ; 0506
                                4F  13 00074        BEQL    11$
                50      20  A7    9A  50 00076      MOVZBL  32(IDX_DFN), R0         ; 0510
                    50          50  01 78 0007A    ASHL    #1, R0, R0
                50              54  D1 0007E        CMPL    R4, R0
                                05  12 00081        BNEQ    6$
                52              56  D0 00083        MOVL    REC_ADDR, REND          ; 0512
                                89  11 00086 5$:    BRB     1$
                52      20      A7  9A 00088 6$:    MOVZBL  32(IDX_DFN), REND       ; 0514
                52              56  C0 0008C        ADDL2   REC_ADDR, REND
                                80  11 0008F        BRB     1$                      ; 0496
                50      20      A7  9A 00091 7$:    MOVZBL  32(IDX_DFN), R0         ; 0525
                56              50  C0 00095        ADDL2   R0, REC_ADDR
                6E              56  D1 00098        CMPL    REC_ADDR, EOB           ; 0527
                                1E  12 0009B        BNEQ    9$
                5B              01  D0 0009D        MOVL    #1, STATUS              ; 0529
                                23  11 000A0        BRB     11$
                50      20      A7  9A 000A2 8$:    MOVZBL  32(IDX_DFN), R0         ; 0540
                56              50  C0 000A6        ADDL2   R0, REC_ADDR
                50      20      A7  9A 000A9        MOVZBL  32(IDX_DFN), R0         ; 0542
                    50          50  01 78 000AD    ASHL    #1, R0, R0
                50              54  D1 000B1        CMPL    R4, R0
                                0A  14 000B4        BGTR    10$
                6E              52  D1 000B6        CMPL    REND, EOB               ; 0550
                                0A  13 000B9        BEQL    11$
                5B              01  CE 000BB 9$:    MNEGL   #1, STATUS              ; 0560
                                05  11 000BE        BRB     11$                     ; 0559
                58              56  D0 000C0 10$:   MOVL    REC_ADDR, RSTART        ; 0566
                                C1  11 000C3        BRB     5$                      ; 0455
                    50          56  55 C3 000C5 11$:   SUBL3   BKT_ADDR, REC_ADDR, R0 ; 0575
                                50  0E C2 000C9    SUBL2   #14, R0
                                51      20  A7  9A 000CC  MOVZBL  32(IDX_DFN), R1
                      0094  C9          50  51 C7 000D0  DIVL3   R1, R0, 148(IRAB)
                                50  5B D0 000D6        MOVL    STATUS, R0              ; 0577
                                5E  04 C0 000D9        ADDL2   #4, SP                  ; 0578
                                    091C  8F  BA 000DC  POPR    #^M<R2,R3,R4,R8,R11>
                                          05 000E0      RSB

; Routine Size:  225 bytes,    Routine Base:  RM$RMS3 + 0000

;  516          0579  1
```

```
;  518          0580  1 ROUTINE SEARCH_V2 (GOAL) : RL$RABREG_567 =
;  519          0581  1 !+++
;  520          0582  1 !
;  521          0583  1 ! SEARCH_V2
;  522          0584  1 !
;  523          0585  1 !       This routine searches a prologue two bucket or a prologue 3 data level
;  524          0586  1 !       bucket (primary or SIDR) with non-compressed keys for a record whose
;  525          0587  1 !       value is equal or greater than the search key.
;  526          0588  1 !
;  527          0589  1 ! INPUT PARAMETER
;  528          0590  1 !
;  529          0591  1 !       GOAL    - 0, return if you find an equal or greater than match
;  530          0592  1 !               - 1, return only with a greater than match
;  531          0593  1 !
;  532          0594  1 ! IMPLICIT INPUTS:
;  533          0595  1 !       REC_ADDR                - address of record in bucket to begin search on
;  534          0596  1 !       BKT_ADDR                - address of current bucket
;  535          0597  1 !       IDX_DFN                 - address of index descriptor for current key of reference
;  536          0598  1 !       IRAB                    - address of internal RAB
;  537          0599  1 !       IRAB[IRB$V_SRCHGT]      - if set, search for index/data record gt search key
;  538          0600  1 !       IRAB[IRB$V_POSINSERT]   - if set, search for position to insert record
;  539          0601  1 !       IRAB[KEY BUFFER 2]      - address of search key
;  540          0602  1 !       IRAB[IRB$B_KEYSZ]       - size of key to compare
;  541          0603  1 !       IFAB[IFB$B_PLG_VER]     - prologue version number
;  542          0604  1 !
;  543          0605  1 ! OUTPUT PARAMETERS:
;  544          0606  1 !       NONE
;  545          0607  1 !
;  546          0608  1 ! IMPLICIT OUTPUTS:
;  547          0609  1 !       REC_ADDR        - if EQ, address of index/record equal to search key
;  548          0610  1 !                       - if GT, at end of record data in bucket
;  549          0611  1 !                       - if LS, address of index/record greater than search key
;  550          0612  1 !       [ DUPS_SEEN ]   - set if duplicates seen when SRCHGT set
;  551          0613  1 !       [ DUP_KEY ]     - set if a duplicate key is seen when SRCHGT is set
;  552          0614  1 !       IRB$L_LST_NCMP  - Address of last record in bucket if search key > data record
;  553          0615  1 !
;  554          0616  1 !
;  555          0617  1 ! ROUTINE VALUE:
;  556          0618  1 !       R0              - 0, search key = index/data record
;  557          0619  1 !                       - -1, search key < index/data record
;  558          0620  1 !                       - 1,  search key > index/data record
;  559          0621  1 !
;  560          0622  1 ! SIDE EFFECTS:
;  561          0623  1 !       RRV are skipped
;  562          0624  1 !       AP is clobbered
;  563          0625  1 !
;  564          0626  1 !--
;  565          0627  2 BEGIN
;  566          0628  2
;  567          0629  2 EXTERNAL REGISTER
;  568          0630  2     R_IRAB_STR,
;  569          0631  2     R_IFAB_STR,
;  570          0632  2     R_IDX_DFN_STR,
;  571          0633  2     R_BKT_ADDR_STR,
;  572          0634  2     R_REC_ADDR_STR;
;  573          0635  2
;  574          0636  2 BUILTIN
```

```
;  575          0637  2      AP;
;  576          0638  2
;  577          0639  2  LOCAL
;  578          0640  2      OVHD,
;  579          0641  2      TOTAL_SIZE,
;  580          0642  2      STATUS,
;  581          0643  2      EOB;
;  582          0644  2
;  583          0645  2  EOB = .BKT_ADDR + .BKT_ADDR [BKT$W_FREESPACE];
;  584          0646  2
;  585          0647  2  WHILE .REC_ADDR LSSA .EOB
;  586          0648  2  DO
;  587          0649  3      BEGIN
;  588          0650  3
;  589          0651  3      AP = 3;  ! Initialize for a contiguous compare
;  590          0652  3
;  591          0653  4      BEGIN
;  592          0654  4
;  593          0655  4      LOCAL
;  594          0656  4          REC_SIZE;
;  595          0657  4
;  596          0658  4      ! Search for the 1st record whose key is greater than or
;  597          0659  4      ! equal to the search key
;  598          0660  4      !
;  599          0661  4
;  600          0662  4      IF (REC_SIZE = .BKT_ADDR [BKT$B_LEVEL]) EQL 0
;  601          0663  4      THEN
;  602          0664  4
;  603          0665  4          ! We have a data record, check if it is an RRV
;  604          0666  4          !
;  605          0667  5          BEGIN
;  606          0668  5
;  607          0669  5          IF .BKT_ADDR[BKT$B_INDEXNO] EQLU 0
;  608          0670  5              AND
;  609          0671  5              .REC_ADDR [IRC$V_RRV]
;  610          0672  5          THEN
;  611          0673  5              RETURN GT;
;  612          0674  5
;  613          0675  5          ! Nope. Do the setup for compare
;  614          0676  5          !
;  615          0677  5
;  616          0678  5          IF .IDX_DFN [IDX$B_KEYREF] NEQ 0
;  617          0679  5          THEN
;  618          0680  5
;  619          0681  5              ! We have a SIDR data record, set REC_SIZE to -1 as flag
;  620          0682  5              !
;  621          0683  5              REC_SIZE = .REC_SIZE - 1
;  622          0684  5
;  623          0685  5          ELSE
;  624          0686  5
;  625          0687  5              ! We have a primary data record, set AP to 2 as flag
;  626          0688  5              !
;  627          0689  5              AP = .AP - 1;
;  628          0690  4          END;
;  629          0691  4
;  630          0692  4      ! Now get the number of bytes of overhead for this record
;  631          0693  4      !
```

```
;  632     0694  4     OVHD = RM$REC_OVHD(.REC_SIZE; REC_SIZE);
;  633     0695  4     TOTAL_SIZE = .OVHD + .REC_SIZE;
;  634     0696  3     END; ! of block defining rec_size
;  635     0697  3
;  636     0698  3     ! Do the comparison (finally).  But first set AP to contiguous compare
;  637     0699  3     ! again, since the primary key is extracted at the front of the record.
;  638     0700  3     !
;  639     0701  3
;  640     0702  3     IF .IFAB[IFB$B_PLG_VER] EQLU PLG$C_VER_3
;  641     0703  3     THEN
;  642     0704  3         AP = 3;
;  643     0705  3
;  644     0706  3     STATUS = RM$COMPARE_KEY (.OVHD + .REC_ADDR,
;  645     0707  3                              KEYBUF_ADDR(2),
;  646     0708  3                              .IRAB [IRB$B_KEYSZ]);
;  647     0709  3
;  648     0710  3     IF .STATUS LSS 0
;  649     0711  3     THEN
;  650     0712  3         RETURN .STATUS;
;  651     0713  3
;  652     0714  3     IF .STATUS EQL 0
;  653     0715  3     THEN
;  654     0716  3         IF .GOAL EQL 0
;  655     0717  3         THEN
;  656     0718  3             RETURN EQ
;  657     0719  3         ELSE
;  658     0720  3             ! We have an equal match but were looking for a GT match.
;  659     0721  3             ! Go get the next record and continue comparison (this will
;  660     0722  3             ! position past all duplicates).
;  661     0723  3             !
;  662     0724  4             BEGIN
;  663     0725  4             IRAB[IRB$V_DUP_KEY] = 1;
;  664     0726  4             IRAB [IRB$L_LST_REC] = .REC_ADDR;
;  665     0727  4
;  666     0728  5             IF  (.BKT_ADDR[BKT$B_LEVEL] EQLU 0
;  667     0729  5                         AND
;  668     0730  5                        .BKT_ADDR[BKT$B_INDEXNO] NEQU 0)
;  669     0731  4                  OR
;  670     0732  5                 NOT (.REC_ADDR[IRC$V_DELETED]
;  671     0733  5                       OR
;  672     0734  5                      .REC_ADDR[IRC$V_RU_DELETE])
;  673     0735  4             THEN
;  674     0736  4                 IRAB [IRB$V_DUPS_SEEN] = 1;
;  675     0737  3             END;
;  676     0738  3
;  677     0739  3         REC_ADDR = .REC_ADDR + .TOTAL_SIZE;
;  678     0740  2         END;
;  679     0741  2
;  680     0742  2 ! We have a GT match. Save the address of the last record in
;  681     0743  2 ! the bucket (we are positioned past it now).
;  682     0744  2 !
;  683     0745  2 IRAB [IRB$L_LST_NCMP] = .REC_ADDR - .TOTAL_SIZE;
;  684     0746  2
;  685     0747  2 RETURN GT
;  686     0748  1 END;
```

```
                091C   8F   BB 00000 SEARCH_V2:
                                             PUSHR   #^M<R2,R3,R4,R8,R11>                    ; 0580
                  5B     04   A5   3C 00004          MOVZWL  4(BKT_ADDR), EOB                ; 0645
                  5B          55   C0 00008          ADDL2   BKT_ADDR, EOB
                  5B          56   D1 0000B 1$:      CMPL    REC_ADDR, EOB                   ; 0647
                              7B   1E 0000E          BGEQU   10$
                  5C          03   D0 00010          MOVL    #3, AP                          ; 0651
                  51     0C   A5   9A 00013          MOVZBL  12(BKT_ADDR), REC_SIZE          ; 0662
                              14   12 00017          BNEQ    4$
                         01   A5   95 00019          TSTB    1(BKT_ADDR)                     ; 0669
                              04   12 0001C          BNEQ    2$
                6F    66          03   E0 0001E          BBS     #3, (REC_ADDR), 11$             ; 0671
                         21   A7   95 00022 2$:      TSTB    33(IDX_DFN)                     ; 0678
                              04   13 00025          BEQL    3$
                              51   D7 00027          DECL    REC_SIZE                        ; 0683
                              02   11 00029          BRB     4$
                              5C   D7 0002B 3$:      DECL    AP                              ; 0689
                           0000G   30 0002D 4$:      BSBW    RM$REC_OVHD                     ; 0694
                  52          50   D0 00030          MOVL    R0, OVHD
                  58      52  51   C1 00033          ADDL3   REC_SIZE, OVHD, TOTAL_SIZE      ; 0695
                  03    00B7  CA   91 00037          CMPB    183(IFAB), #3                   ; 0702
                              03   12 0003C          BNEQ    5$
                  5C          03   D0 0003E          MOVL    #3, AP                          ; 0704
                  53    00B4  CA   3C 00041 5$:      MOVZWL  180(IFAB), R3                   ; 0707
                  53     60   A9   C0 00046          ADDL2   96(IRAB), R3
                  51      52  56   C1 0004A          ADDL3   REC_ADDR, OVHD, R1              ; 0706
                  50    00A6  C9   9A 0004E          MOVZBL  166(IRAB), R0
                           0000G   30 00053          BSBW    RM$COMPARE_KEY
                  54          50   D0 00056          MOVL    R0, STATUS
                              05   18 00059          BGEQ    6$                              ; 0710
                  50          54   D0 0005B          MOVL    STATUS, R0                      ; 0712
                              38   11 0005E          BRB     13$
                              24   12 00060 6$:      BNEQ    9$                              ; 0714
                        18    AE   D5 00062          TSTL    GOAL                            ; 0716
                              2F   13 00065          BEQL    12$
                  43    A9          01   88 00067          BISB2   #1, 67(IRAB)                    ; 0725
                  4C    A9          56   D0 0006B          MOVL    REC_ADDR, 76(IRAB)              ; 0726
                        0C    A5   95 0006F          TSTB    12(BKT_ADDR)                    ; 0728
                              05   12 00072          BNEQ    7$
                        01    A5   95 00074          TSTB    1(BKT_ADDR)                     ; 0730
                              08   12 00077          BNEQ    8$
                09            66   02   E0 00079 7$:      BBS     #2, (REC_ADDR), 9$              ; 0732
                05            66   05   E0 0007D          BBS     #5, (REC_ADDR), 9$              ; 0734
                  44    A9    80   8F   88 00081 8$:      BISB2   #128, 68(IRAB)                  ; 0736
                              56   58   C0 00086 9$:      ADDL2   TOTAL_SIZE, REC_ADDR            ; 0739
                              80   11 00089          BRB     1$                              ; 0647
              0098    C9          56   58   C3 0008B 10$:     SUBL3   TOTAL_SIZE, REC_ADDR, 152(IRAB) ; 0745
                              50   01   D0 00091 11$:     MOVL    #1, R0                          ; 0747
                              02   11 00094          BRB     13$
                              50   D4 00096 12$:      CLRL    R0                              ; 0748
                        091C  8F   BA 00098 13$:      POPR    #^M<R2,R3,R4,R8,R11>
                              05   0009C          RSB

; Routine Size:  157 bytes,    Routine Base:  RM$RMS3 + 00E1
```

```
;  688          0749  1 GLOBAL ROUTINE RM$SRCH_BY_KEY : RL$RABREG_567 =
;  689          0750  1 !++
;  690          0751  1 !
;  691          0752  1 !    RM$SRCH_BY_KEY
;  692          0753  1 !
;  693          0754  1 !       This routine searches a bucket from the current record address
;  694          0755  1 !       for an index/data record equal or greater than the input search key
;  695          0756  1 !
;  696          0757  1 ! CALLING SEQUENCE:
;  697          0758  1 !       RM$SRCH_BY_KEY()
;  698          0759  1 !
;  699          0760  1 ! INPUT PARAMETERS:
;  700          0761  1 !       NONE
;  701          0762  1 !
;  702          0763  1 ! IMPLICIT INPUTS:
;  703          0764  1 !       REC_ADDR                - address of record in bucket to begin search on
;  704          0765  1 !       BKT_ADDR                - address of current bucket
;  705          0766  1 !       IDX_DFN                 - address of index descriptor for current key of reference
;  706          0767  1 !       IRAB                    - address of internal RAB
;  707          0768  1 !       IRAB[IRB$V_SRCHGT]      - if set, search for index/data record gt search key
;  708          0769  1 !       IRAB[IRB$V_POSINSERT]   - if set, search for position to insert record
;  709          0770  1 !       IRAB[KEY_BUFFER_2]      - address of search key
;  710          0771  1 !       IRAB[IRB$B_KEYSZ]       - size of key to compare
;  711          0772  1 !
;  712          0773  1 ! OUTPUT PARAMETERS:
;  713          0774  1 !       NONE
;  714          0775  1 !
;  715          0776  1 ! IMPLICIT OUTPUTS:
;  716          0777  1 !       REC_ADDR        - if EQ, address of index/record equal to search key
;  717          0778  1 !                       - if GT, at end of record data in bucket
;  718          0779  1 !                       - if LS, address of index/record greater than search key
;  719          0780  1 !       IRAB [ EMPT_SEEN ] - set if no data records encountered
;  720          0781  1 !            [ DUPS_SEEN ] - set if duplicates seen when SRCHGT set
;  721          0782  1 !            [ DUP_KEY ]   - set if duplicate key seen when SRCHGT is set
;  722          0783  1 !
;  723          0784  1 ! ROUTINE VALUE:
;  724          0785  1 !       R0              - 0, search key = index/data record
;  725          0786  1 !                       - -1, search key < index/data record
;  726          0787  1 !                       - 1,  search key > index/data record
;  727          0788  1 !
;  728          0789  1 ! SIDE EFFECTS:
;  729          0790  1 !       RRV are skipped
;  730          0791  1 !       AP is clobbered
;  731          0792  1 !
;  732          0793  1 !--
;  733          0794  1
;  734          0795  2     BEGIN
;  735          0796  2
;  736          0797  2     EXTERNAL REGISTER
;  737          0798  2         COMMON_RAB_STR,
;  738          0799  2         R_BKT_ADDR_STR,
;  739          0800  2         R_REC_ADDR_STR,
;  740          0801  2         R_IDX_DFN_STR;
;  741          0802  2
;  742          0803  2     LOCAL
;  743          0804  2         BKTYP,                  ! Type of bucket we are searching
;  744          0805  2         GOAL,                   ! Flag to say 'search past dups'
```

```
;  745          0806  2          STATUS;                     ! Result of a given compare
;  746          0807  2
;  747          0808  2
;  748          0809  2      ! If the record we are about to look at is an RRV, then we have an
;  749          0810  2      ! empty bucket.
;  750          0811  2      !
;  751          0812  2      IF   .BKT_ADDR[BKT$B_LEVEL] EQL 0
;  752          0813  2           AND
;  753          0814  2           .BKT_ADDR[BKT$B_INDEXNO] EQLU 0
;  754          0815  2      THEN
;  755          0816  3          BEGIN
;  756          0817  3          LOCAL
;  757          0818  3              CNTRL: REF BBLOCK;  ! Control byte for 1st record
;  758          0819  3
;  759          0820  3          CNTRL = RM$CNTRL_ADDR(0);
;  760          0821  3
;  761          0822  3          IF .CNTRL [IRC$V_RRV]
;  762          0823  3          THEN
;  763          0824  4              BEGIN
;  764          0825  4              IRAB [IRB$V_EMPT_SEEN] = 1;
;  765          0826  4              RETURN GT
;  766          0827  3              END;
;  767          0828  2          END;
;  768          0829  2
;  769          0830  2      ! Should we be satisfied with an equal match or continue searching
;  770          0831  2      ! for a greater than match ?
;  771          0832  2      !
;  772          0833  2      GOAL = EQ;
;  773          0834  2
;  774          0835  2      IF .IRAB [IRB$V_SRCHGT]
;  775          0836  3          OR (.IRAB [IRB$V_POSINSERT] AND .BKT_ADDR [BKT$B_LEVEL] EQL 0)
;  776          0837  2      THEN
;  777          0838  2          GOAL = GT;
;  778          0839  2
;  779          0840  2      ! Now actually search the !!!!aa## bucket
;  780          0841  2      ! First find out what kind of bucket it is
;  781          0842  2      !
;  782          0843  2
;  783          0844  2      IF .BKT_ADDR [BKT$B_LEVEL] EQL 0
;  784          0845  2      THEN
;  785          0846  2          BKTYP = .IDX_DFN [IDX$B_DATBKTYP]
;  786          0847  2      ELSE
;  787          0848  2          BKTYP = .IDX_DFN [IDX$B_IDXBKTYP];
;  788          0849  2
;  789          0850  2      CASE .BKTYP FROM IDX$C_V2_BKT TO IDX$C_NCMPNCMP OF
;  790          0851  2          SET
;  791          0852  2          [IDX$C_V2_BKT]: ! Prologue two index bucket.
;  792          0853  2                          !
;  793          0854  2                          STATUS = SEARCH_V2 (.GOAL);
;  794          0855  2
;  795          0856  2          [IDX$C_CMPIDX]: ! Prologue three compressed index bucket.
;  796          0857  2                          !
;  797          0858  2                          STATUS = RM$SRCH_CMPR (.GOAL);
;  798          0859  2
;  799          0860  2          [IDX$C_NCMPIDX]: ! Prologue three noncompressed index bucket.
;  800          0861  2                          !
;  801          0862  2                          STATUS = SEARCH_FIX (.GOAL);
```

```
;  802          0863  2
;  803          0864  2          [IDX$C_CMPCMP]: ! Prologue three primary data bucket.
;  804          0865  2                          ! Primary key is compressed, data is compressed.
;  805          0866  2                          !
;  806          0867  2                          ! Prologue 3 compressed SIDR bucket.
;  807          0868  2                          !
;  808          0869  2                          STATUS = RM$SRCH_CMPR (.GOAL);
;  809          0870  2
;  810          0871  2          [IDX$C_CMPNCMP]: ! Prologue three primary data bucket.
;  811          0872  2                          ! Primary key is compressed, data is NOT compressed
;  812          0873  2                          !
;  813          0874  2                          STATUS = RM$SRCH_CMPR (.GOAL);
;  814          0875  2
;  815          0876  2          [IDX$C_NCMPCMP]: ! Prologue three data bucket.
;  816          0877  2                          ! Primary key is NOT compressed, data is compressed
;  817          0878  2                          !
;  818          0879  2                          ! Prologue 3 noncompressed SIDR bucket.
;  819          0880  2                          !
;  820          0881  2                          STATUS = SEARCH_V2 (.GOAL);
;  821          0882  2
;  822          0883  2          [IDX$C_NCMPNCMP]: ! Prologue three data bucket.
;  823          0884  2                          ! Primary key is NOT compressed, data is NOT compressed
;  824          0885  2                          !
;  825          0886  2                          STATUS = SEARCH_V2 (.GOAL);
;  826          0887  2          TES;
;  827          0888  2
;  828          0889  2      RETURN .STATUS
;  829          0890  1      END;


                                0C    A5  95 00000 RM$SRCH_BY_KEY::
                                                   TSTB    12(BKT_ADDR)                    ; 0812
                                      19  12 00003        BNEQ    1$
                                01    A5  95 00005        TSTB    1(BKT_ADDR)             ; 0814
                                      14  12 00008        BNEQ    1$
                                      7E  D4 0000A        CLRL    -(SP)                   ; 0820
                                   0000G  30 0000C        BSBW    RM$CNTRL_ADDR
                             5E         04  C0 0000F        ADDL2   #4, SP
                          08 60         03  E1 00012        BBC     #3, (CNTRL), 1$         ; 0822
                       44 A9            02  88 00016        BISB2   #2, 68(IRAB)            ; 0825
                             50         01  D0 0001A        MOVL    #1, R0                  ; 0826
                                        05  00 0001D        RSB
                                        51  D4 0001E 1$:     CLRL    GOAL                    ; 0833
                          09 42 A9      01  E0 00020        BBS     #1, 66(IRAB), 2$        ; 0835
                             08         42  A9 E9 00025     BLBC    66(IRAB), 3$            ; 0836
                                   0C   A5  95 00029        TSTB    12(BKT_ADDR)
                                        03  12 0002C        BNEQ    3$
                             51         01  D0 0002E 2$:     MOVL    #1, GOAL                ; 0838
                                   0C   A5  95 00031 3$:     TSTB    12(BKT_ADDR)            ; 0844
                                        06  12 00034        BNEQ    4$
                             50    29   A7  9A 00036        MOVZBL  41(IDX_DFN), BKTYP      ; 0846
                                        04  11 0003A        BRB     5$
                             50    28   A7  9A 0003C 4$:     MOVZBL  40(IDX_DFN), BKTYP      ; 0848
                          06 00         50  CF 00040 5$:     CASEL   BKTYP, #0, #6           ; 0850
```

```
        0010            0017            0010            001E     00044 6$:      .WORD   9$-6$,-                                 ;
                        001E            001E            0010     0004C                  7$-6$,-                                 ;
                                                                                       8$-6$,-                                 ;
                                                                                       7$-6$,-                                 ;
                                                                                       7$-6$,-                                 ;
                                                                                       9$-6$,-                                 ;
                                                                                       9$-6$                                   ;
                                                          0E  11 00052                 BRB     9$                              ; 0854
                                                          51  DD 00054 7$:             PUSHL   GOAL                            ; 0858
                                                       0000G  30 00056                 BSBW    RM$SRCH_CMPR                    ;
                                                          0C  11 00059                 BRB     10$                             ;
                                                          51  DD 0005B 8$:             PUSHL   GOAL                            ; 0862
                                                        FE22  30 0005D                 BSBW    SEARCH_FIX                      ;
                                                          05  11 00060                 BRB     10$                             ;
                                                          51  DD 00062 9$:             PUSHL   GOAL                            ; 0886
                                                        FEFC  30 00064                 BSBW    SEARCH_V2                       ;
                                        5E                04  C0 00067 10$:           ADDL2   #4, SP                          ;
                                                              05 0006A                 RSB                                     ; 0890

; Routine Size:  107 bytes,    Routine Base:  RM$RMS3 + 017E

;  830          0891  1
```

```
;  832     0892  1 ROUTINE RM$SEARCH_TREE : RL$RABREG_4567 =
;  833     0893  1
;  834     0894  1 !++
;  835     0895  1 !
;  836     0896  1 ! FUNCTIONAL DESCRIPTION:
;  837     0897  1 !
;  838     0898  1 !       This routine searches from the current record in the current bucket
;  839     0899  1 !       to the stop level requested for a data record /index equal to or
;  840     0900  1 !       greater than the search key.
;  841     0901  1 !       NOTE: this routine should never be called by an outside routine
;  842     0902  1 !               RM$CSEARCH_TREE should be used in its place
;  843     0903  1 !
;  844     0904  1 ! CALLING SEQUENCE:
;  845     0905  1 !       RM$SEARCH_TREE()
;  846     0906  1 !
;  847     0907  1 ! INPUT PARAMETERS:
;  848     0908  1 !       NONE
;  849     0909  1 !
;  850     0910  1 ! IMPLICIT INPUTS:
;  851     0911  1 !       BKT_ADDR                - address of current bucket
;  852     0912  1 !       REC_ADDR                - address of record in bucket to start search at
;  853     0913  1 !       IDX_DFN                 - address of index descriptor for current key of reference
;  854     0914  1 !       IRAB                    - address of internal RAB
;  855     0915  1 !       IRAB[KEYBUF2]           - address of search key
;  856     0916  1 !       IRAB[IRB$B_KYSZ]        - size of key to compare(not equal to key size if generic search)
;  857     0917  1 !       IRAB[IRB$B_STOPLEVEL]   - level to stop search at
;  858     0918  1 !       IRAB[IRB$W_SRCHFLAGS]
;  859     0919  1 !       IRAB[IRB$V_POSINSERT]   - if set, this is a position for insert
;  860     0920  1 !       IRAB[IRB$V_POSDELETE]   - if set, this is position for delete
;  861     0921  1 !       IRAB[IRB$V_SRCHGT]      - if set, this is a GT approximate search
;  862     0922  1 !       IRAB[IRB$V_SRCHGE]      - if set, this is a GE approximate search
;  863     0923  1 !       IRAB[IRB$V_FIRST_TIM]   - if set, this is the first seq. positioning
;  864     0924  1 !                                 - after a $connect or $rewind
;  865     0925  1 !       IFAB[IFB$B_EXTRABUF]    - used to decide whether to try lockabove
;  866     0926  1 !                                 performance optimization coming down tree
;  867     0927  1 !       IFAB[IFB$V_RU]          - if set, file is RU Journallable
;  868     0928  1 !       IFAB[IFB$V_WRTACC]      - if set, file is write accessed
;  869     0929  1 !
;  870     0930  1 ! OUTPUT PARAMETERS:
;  871     0931  1 !       NONE
;  872     0932  1 !
;  873     0933  1 ! IMPLICIT OUTPUTS:
;  874     0934  1 !       REC_ADDR                - address of index/data record which terminated search
;  875     0935  1 !       IRAB[IRB$L_CURBDB]      - address of current BDB
;  876     0936  1 !       BDB                     - address of current BDB
;  877     0937  1 !       IRAB[IRB$L_LOCK_BDB]    - address of level above current if locked
;  878     0938  1 !       IRAB[IRB$V_ABOVELCKD]   - set when level above data level locked
;  879     0939  1 !       IRAB[IRB$V_DUPS_SEEN]   - set if non-deleted duplicate key encountered
;  880     0940  1 !                                 at data level on position for insert
;  881     0941  1 !       IRAB[IRB$V_DUP_KEY]     - set if duplicate key encountered at data level
;  882     0942  1 !                                 on position for insert
;  883     0943  1 !       IRAB [IRB$L_REC_COUNT]  - Set to the number of the record we found in
;  884     0944  1 !                                 a prologue three bucket
;  885     0945  1 !
;  886     0946  1 ! ROUTINE VALUE:
;  887     0947  1 !       RMS$_RNF                - record not found
;  888     0948  1 !       RMS$_SUC                - record found, in approximate search key
```

```
;  889          0949  1 !                                       may not equal record/index key
;  890          0950  1 !       RMS$_RLK                      - on an horizontal search at level zero for position
;  891          0951  1 !                                       for insert, a lock error was encountered
;  892          0952  1 !       miscellaneous I/O errors
;  893          0953  1 !
;  894          0954  1 ! SIDE EFFECTS:
;  895          0955  1 !       IRAB [ EMPT_SEEN ]          - may be clobbered at any level
;  896          0956  1 !            [ EMPTY_BKT ]          - may be clobbered at data level
;  897          0957  1 !
;  898          0958  1 !--
;  899          0959  1
;  900          0960  2     BEGIN
;  901          0961  2
;  902          0962  2     LABEL
;  903          0963  2         BLK,
;  904          0964  2         BLK1,
;  905          0965  2         BLK2,
;  906          0966  2         BLK3,
;  907          0967  2         BLK4,
;  908          0968  2         BLK5,
;  909          0969  2         BLK6;
;  910          0970  2
;  911          0971  2     BUILTIN
;  912          0972  2         AP,
;  913          0973  2         TESTBITSC;
;  914          0974  2
;  915          0975  2     LOCAL
;  916          0976  2         VBN;
;  917          0977  2
;  918          0978  2     EXTERNAL REGISTER
;  919          0979  2         COMMON_RAB_STR,
;  920          0980  2         COMMON_IO_STR,
;  921          0981  2         R_REC_ADDR_STR,
;  922          0982  2         R_IDX_DFN_STR;
;  923          0983  2
;  924          0984  2     VBN = 0;
;  925          0985  2
;  926          0986  2     DO
;  927          0987  2 BLK2 :
;  928          0988  3         BEGIN
;  929          0989  3
;  930          0990  3         LOCAL
;  931          0991  3             ST;
;  932          0992  3
;  933          0993  3         ST = 0;
;  934          0994  3
;  935          0995  3         ! if this is the 1st time, just get the 1st down pointer w/o searching
;  936          0996  3         !
;  937          0997  3
;  938          0998  3         IF NOT .IRAB[IRB$V_FIRST_TIM]
;  939          0999  3         THEN
;  940          1000  3             ST = RM$SRCH_BY_KEY()
;  941          1001  3         ELSE
;  942          1002  3             IRAB [IRB$L_LST_NCMP] = .BKT_ADDR + BKT$C_OVERHDSZ;
;  943          1003  3
;  944          1004  3         ! If the status is GT then no record in this bucket terminated the
;  945          1005  3         ! search. Therefore search horizontally for termination record.
```

```
;  946          1006  3          ! Also search horiz when skipping over dups on position for insert
;  947          1007  3          !
;  948          1008  3          !
;  949          1009  3
;  950          1010  3          IF .ST GTR 0
;  951          1011  3          THEN
;  952          1012  4              BEGIN
;  953          1013  4
;  954          1014  4              LOCAL
;  955          1015  4                  SIZE;
;  956          1016  4
;  957          1017  4              IF .BKT_ADDR[BKT$V_LASTBKT]
;  958          1018  4              THEN
;  959          1019  4
;  960          1020  4                  IF .IRAB[IRB$V_POSINSERT]
;  961          1021  4                  THEN
;  962          1022  5                      RETURN RMSSUC(SUC)
;  963          1023  4                  ELSE
;  964          1024  4                      RETURN RMSERR(RNF);
;  965          1025  4
;  966          1026  4              BDB = .IRAB[IRB$L_CURBDB];
;  967          1027  4              SIZE = .BDB[BDB$W_NUMB];
;  968          1028  4
;  969          1029  4              IF .VBN EQL 0
;  970          1030  4              THEN
;  971          1031  4                  VBN = .BKT_ADDR[BKT$L_NXTBKT];
;  972          1032  4
;  973          1033  5              IF (.IRAB[IRB$V_POSINSERT] AND .BKT_ADDR[BKT$B_LEVEL] EQLU 0)
;  974          1034  4              THEN
;  975          1035  5 BLK3 :           BEGIN
;  976          1036  5
;  977          1037  5                  ! Check if we can insert the new key into the left
;  978          1038  5                  !  hand bucket (the one we've got) or if we have to
;  979          1039  5                  !  go get the right hand bucket in order to make the
;  980          1040  5                  !  decision.
;  981          1041  5                  !
;  982          1042  5                  ! NEXT_DOWN will be equal to -1 if none was saved,
;  983          1043  5                  ! causing this test to fail.  Once this test has been
;  984          1044  5                  ! made once, NEXT_DOWN is set to zero, again causing
;  985          1045  5                  ! failure on this test.
;  986          1046  5                  !
;  987          1047  5
;  988          1048  5                  IF .VBN EQLU .IRAB [IRB$L_NEXT_DOWN]
;  989          1049  5                  THEN
;  990          1050  5                      ! Horizontal pointer = down pointer
;  991          1051  5                      !
;  992          1052  5
;  993          1053  6                      IF  NOT (.IDX_DFN [IDX$V_DUPKEYS] AND .IDX_DFN [IDX$B_KEYREF] EQL 0)
;  994          1054  5                      THEN
;  995          1055  5                          ! Not a primary data bucket with possible duplicates
;  996          1056  5                          !
;  997          1057  5                          RETURN RMSSUC(SUC);
;  998          1058  5
;  999          1059  5                  ! If the file is being shared in any way, and the bucket
; 1000          1060  5                  ! in the level above was not locked coming down the tree,
; 1001          1061  5                  ! we must re-access the level above to confirm that in
; 1002          1062  5                  ! fact a split has not occurred and the level above index
```

```
;1003     1063  5          ! bucket doesn't have a new key value in the pointer to
;1004     1064  5          ! the bucket we are in.  If this is not done, the horizontal
;1005     1065  5          ! positioning logic that follows may cause the record to
;1006     1066  5          ! be inserted such that it is not accessible by random access.
;1007     1067  5          ! The following pictures illustrate.  Assume 1 record per
;1008     1068  5          ! bucket at the data level.
;1009     1069  5          !
;1010     1070  5          !   ----
;1011     1071  5          !  | 9  |  level 1    Index looks like this when coming
;1012     1072  5          !   ----               down tree looking to insert an "8".
;1013     1073  5          !    v
;1014     1074  5          !
;1015     1075  5          !  |-9--|  data level
;1016     1076  5          !
;1017     1077  5          !   db-1
;1018     1078  5          !
;1019     1079  5          !  |5-9-|       However, by the time that db 1 is actually
;1020     1080  5          !   ----         accessed, it has split and the index updated.
;1021     1081  5          !  /    \        Yet, because the key value "8" is less than the
;1022     1082  5          ! ---  ---       lowest key value in db 2, the position for
;1023     1083  5          !| 5 || 9 |      insert would be in db 1.  Rescanning the level
;1024     1084  5          ! ---  ---       1 bucket for key "8" will now find the correct
;1025     1085  5          !db 1  db 2      down pointer to db 2.
;1026     1086  5          !
;1027     1087  5
;1028     1088  5          ! If the down pointer from level 1 can not have changed,
;1029     1089  5          ! either because the file is not shared or it is still
;1030     1090  5          ! locked, or it has already been checked once, then the
;1031     1091  5          ! check does not have to be made.
;1032     1092  5          !
;1033     1093  5
;1034     1094  5          IF .IFAB [IFB$V_NORECLK]
;1035     1095  5              OR .IRAB [IRB$V_ABOVELCKD]
;1036     1096  5          THEN
;1037     1097  6              BEGIN
;1038     1098  6              IF .IRAB [IRB$L_NEXT_DOWN] NEQ 0
;1039     1099  6              THEN
;1040     1100  7                  BEGIN
;1041     1101  7                  IRAB [IRB$L_VBN_RIGHT] = 0;
;1042     1102  7                  IRAB [IRB$L_NEXT_DOWN] = 0;
;1043     1103  7                  IRAB [IRB$L_VBN_LEFT] = .BDB [BDB$L_VBN];
;1044     1104  6                  END;
;1045     1105  6
;1046     1106  6              LEAVE BLK3;
;1047     1107  6              END
;1048     1108  5          ELSE
;1049     1109  5
;1050     1110  5              ! If NEXT_DOWN is zero, we've already been through
;1051     1111  5              ! here once, so don't check again.
;1052     1112  5              !
;1053     1113  5
;1054     1114  5              IF .IRAB [IRB$L_NEXT_DOWN] EQL 0
;1055     1115  5              THEN
;1056     1116  5                  LEAVE BLK3;
;1057     1117  5
;1058     1118  5
;1059     1119  5          ! The VBN of the level 1 bucket has been saved in the
```

```
;  1060          1120  5                ! VBN_RIGHT field.  Reaccess the bucket.  Release the
;  1061          1121  5                ! level 0 bucket and exit on errors.
;  1062          1122  5                !
;  1063          1123  5
;  1064        P 1124  5                ST = CACHE (.IRAB [IRB$L_VBN_RIGHT],
;  1065          1125  5                            .IDX_DFN [IDX$B_IDXBKTSZ]*512);
;  1066          1126  5                IF NOT .ST
;  1067          1127  5                THEN
;  1068          1128  6                    BEGIN
;  1069          1129  6                    BDB = .IRAB [IRB$L_CURBDB];
;  1070          1130  6                    RM$RLSBKT(0);
;  1071          1131  6                    IRAB [IRB$L_CURBDB] = 0;
;  1072          1132  6                    RETURN .ST;
;  1073          1133  5                    END;
;  1074          1134  5
;  1075          1135  5                ! Rescan the level 1 bucket just accessed.  If the
;  1076          1136  5                ! key value is no longer in the bucket at all, then
;  1077          1137  5                ! release both level 1 and level 0 buckets and come
;  1078          1138  5                ! down the tree from the top (returning RLK to CSEARCH_TREE
;  1079          1139  5                ! does that).
;  1080          1140  5                !
;  1081          1141  5
;  1082          1142  5                REC_ADDR = .BKT_ADDR + BKT$C_OVERHDSZ;
;  1083          1143  5                IRAB [IRB$L_REC_COUNT] = 0;
;  1084          1144  5                IF RM$SRCH_BY_KEY() GTR 0
;  1085          1145  5                THEN
;  1086          1146  6                    BEGIN
;  1087          1147  6                    ST = RMSERR(RLK);
;  1088          1148  6                    RM$RLSBKT(0);
;  1089          1149  6                    BDB = .IRAB [IRB$L_CURBDB];
;  1090          1150  6                    RM$RLSBKT(0);
;  1091          1151  6                    IRAB [IRB$L_CURBDB] = 0;
;  1092          1152  6                    RETURN .ST;
;  1093          1153  5                    END;
;  1094          1154  5
;  1095          1155  5                ! At this point we've a down pointer in the same level
;  1096          1156  5                ! 1 bucket.  Check the pointer itself to see if it still
;  1097          1157  5                ! points to the level 0 bucket we found before.
;  1098          1158  5                !
;  1099          1159  5
;  1100          1160  6                BEGIN
;  1101          1161  6
;  1102          1162  6                LOCAL
;  1103          1163  6                    LEV0_BDB    : REF BBLOCK,
;  1104          1164  6                    LEV1_VBN;
;  1105          1165  6
;  1106          1166  6                LEV0_BDB = .IRAB [IRB$L_CURBDB];
;  1107          1167  6                AP = 1;
;  1108          1168  6
;  1109          1169  6                IF .IFAB [IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  1110          1170  6                THEN
;  1111          1171  6                    LEV1_VBN - RM$RECORD_VBN()
;  1112          1172  6                ELSE
;  1113          1173  6                    LEV1_VBN = RM$V3_VBN();
;  1114          1174  6
;  1115          1175  6                IF .LEV1_VBN NEQ  .LEV0_BDB [BDB$L_VBN]
;  1116          1176  6                THEN
```

```
;  1117     1177  6
;  1118     1178  6                    ! The pointer from the level 1 bucket is not the same,
;  1119     1179  6                    ! but it is in the same level 1 bucket.  Simply release
;  1120     1180  6                    ! the level 0 bucket we have accessed, and go around the
;  1121     1181  6                    ! loop again so that this all happens again.
;  1122     1182  6                    !
;  1123     1183  7                    BEGIN
;  1124     1184  7                    IRAB [IRB$B_SPL_BITS] = 0;
;  1125     1185  7                    IRAB [IRB$L_CURBDB] = .BDB;
;  1126     1186  7                    BDB = .LEV0_BDB;
;  1127     1187  7                    RM$RLSBKT(0);
;  1128     1188  7                    BDB = .IRAB [IRB$L_CURBDB];
;  1129     1189  7                    BKT_ADDR = .BDB [BDB$L_ADDR];
;  1130     1190  7                    REC_ADDR = .BKT_ADDR + BKT$C_OVERHDSZ;
;  1131     1191  7                    LEAVE BLK2;
;  1132     1192  6                    END;
;  1133     1193  6
;  1134     1194  6                ! Ok, everything is cool.  The down pointer is still the
;  1135     1195  6                ! same.  Release the level 1 bucket and prepare to follow
;  1136     1196  6                ! the horizontal links at the data level for primary key.
;  1137     1197  6                !
;  1138     1198  6
;  1139     1199  6                RM$RLSBKT(0);
;  1140     1200  6                BDB = .LEV0_BDB;
;  1141     1201  6                BKT_ADDR = .BDB [BDB$L_ADDR];
;  1142     1202  5                END;            ! of block defining LEV* locals.
;  1143     1203  5
;  1144     1204  5                IRAB [IRB$L_VBN_RIGHT] = 0;
;  1145     1205  5                IRAB [IRB$L_NEXT_DOWN] = 0;
;  1146     1206  5                IRAB [IRB$L_VBN_LEFT] = .BDB [BDB$L_VBN];
;  1147     1207  5
;  1148     1208  4                END;
;  1149     1209  4
;  1150     1210  4                ! Under the following circumstances, the bucket to be read must
;  1151     1211  4                ! be locked:
;  1152     1212  4                !
;  1153     1213  4                ! 1. If this is the stoplevel on any positioning for modification.
;  1154     1214  4                ! 2. If this is a SIDR bucket and the file is write accessed.
;  1155     1215  4                ! 3. If this is a primary data bucket, the file is write accessed
;  1156     1216  4                !    and marked RU Journallable.
;  1157     1217  4                !
;  1158     1218  6                IF ((.IRAB[IRB$W_SRCHFLAGS]
;  1159     1219  6                                    AND
;  1160     1220  5                                   (IRB$M_POSINSERT + IRB$M_POSDELETE)) NEQ 0
;  1161     1221  5                             AND
;  1162     1222  5                            .IRAB[IRB$B_STOPLEVEL] EQL .BKT_ADDR[BKT$B_LEVEL])
;  1163     1223  4                  OR
;  1164     1224  6                 ((.IRAB[IRB$W_SRCHFLAGS]
;  1165     1225  6                                    AND
;  1166     1226  5                                   (IRB$M_POSINSERT + IRB$M_POSDELETE)) EQLU 0
;  1167     1227  5                             AND
;  1168     1228  5                            .IFAB[IFB$V_WRTACC]
;  1169     1229  5                             AND
;  1170     1230  5                            .BKT_ADDR[BKT$B_LEVEL] EQLU 0
;  1171     1231  5                             AND
;  1172     1232  6                            (.IFAB[IFB$V_RU]
;  1173     1233  6                                    OR
```

```
;1174     1234  5                                .IDX_DFN[IDX$B_KEYREF] GTRU 0))
;1175     1235  4            THEN
;1176     1236  4                IRAB [IRB$B_CACHEFLGS] = CSH$M_LOCK;
;1177     1237  4
;1178     1238  4            ! Release current bucket before reading the next one.
;1179     1239  4            !
;1180     1240  4
;1181     1241  4            RM$RLSBKT(0);
;1182     1242  4            IRAB [IRB$L_CURBDB] = 0;
;1183     1243  4
;1184     1244  4            ! Read in the next bucket in the horizontal chain.
;1185     1245  4            !
;1186     1246  4
;1187     1247  5            IF NOT (ST = RM$GETBKT(.VBN, .SIZE))
;1188     1248  4            THEN
;1189     1249  5                BEGIN
;1190     1250  5
;1191     1251  5                ! If the level above was locked, release it.
;1192     1252  5                !
;1193     1253  5
;1194     1254  5                IF TESTBITSC(IRAB[IRB$V_ABOVELCKD])
;1195     1255  5                THEN
;1196     1256  5                    RELEASE(IRAB[IRB$L_LOCK_BDB]);
;1197     1257  5
;1198     1258  5                RETURN .ST;
;1199     1259  4                END;
;1200     1260  4
;1201     1261  4            VBN = .BKT_ADDR[BKT$L_NXTBKT];
;1202     1262  4            IRAB[IRB$L_CURBDB] = .BDB;
;1203     1263  4
;1204     1264  5            IF (.IRAB[IRB$V_POSINSERT] AND .BKT_ADDR[BKT$B_LEVEL] EQLU 0)
;1205     1265  4            THEN
;1206     1266  4
;1207     1267  4                ! This is a position for insert at level 0.
;1208     1268  4                !
;1209     1269  4                ! At this point we have just read in the next bucket in the
;1210     1270  4                ! horizontal chain.  The position for insert in the bucket
;1211     1271  4                ! just released was at the end of that bucket.
;1212     1272  4                !
;1213     1273  4                ! Basically, if the record is lower than any record in
;1214     1274  4                ! the next bucket, it should be inserted at the end of
;1215     1275  4                ! the previous bucket.
;1216     1276  4                !
;1217     1277  4                ! Field usage is as follows:
;1218     1278  4                !  IRAB[ VBN_LEFT ] - the leftmost, non-empty bucket in
;1219     1279  4                !       which the position for insert was at the end of
;1220     1280  4                !       the bucket.
;1221     1281  4                !
;1222     1282  4                !  IRAB[ VBN_RIGHT ] - the rightmost bucket in which the
;1223     1283  4                !       position for insert was at the beginning of the
;1224     1284  4                !       bucket, i.e., less than all existing values in
;1225     1285  4                !       that bucket, or an empty bucket.  This is used
;1226     1286  4                !       to determine if VBN_LEFT may be used after backing
;1227     1287  4                !       up.
;1228     1288  4                !
;1229     1289  4                ! Because the buckets are released prior to accessing the
;1230     1290  4                ! next bucket, it is possible for splits to occur prior to
```

```
;  1231          1291  4          ! backing up, if that is necessary.  The following assumptions
;  1232          1292  4          ! are made:
;  1233          1293  4          ! 1) The position for insert can never be to the left of
;  1234          1294  4          ! the initial VBN_LEFT.
;  1235          1295  4          ! 2) The position for insert can never be in or to the right
;  1236          1296  4          ! of a bucket in which the position for insert was at the
;  1237          1297  4          ! beginning of that bucket.
;  1238          1298  4          ! 3) If the NXTBKT link from a VBN_LEFT matches a previously
;  1239          1299  4          ! accessed VBN_RIGHT, the correct position for insert is at
;  1240          1300  4          ! the end of VBN_LEFT.
;  1241          1301  4          ! 4) Empty buckets are skipped over until a non-empty bucket
;  1242          1302  4          ! is encountered.
;  1243          1303  4          !
;  1244          1304  4          ! Note that an EMPT_SEEN bucket also returns greater than (GT)
;  1245          1305  4          ! status.  VBN_LEFT and VBN_RIGHT are initialized to zero
;  1246          1306  4          ! in the CSEARCH_TREE routine when positioning for insert and
;  1247          1307  4          ! the stoplevel is 0.
;  1248          1308  4          !
;  1249          1309  4
;  1250          1310  4          WHILE 1 DO
;  1251          1311  4
;  1252          1312  5          BEGIN
;  1253          1313  5
;  1254          1314  5          REC_ADDR = .BKT_ADDR + BKT$C_OVERHDSZ;
;  1255          1315  5          IRAB [IRB$L_REC_COUNT] = 0;
;  1256          1316  5
;  1257          1317  5          ST = RM$SRCH_BY_KEY();
;  1258          1318  5
;  1259          1319  5          IF (VBN=.BKT_ADDR[BKT$L_NXTBKT]) EQL .IRAB[IRB$L_VBN_RIGHT]
;  1260          1320  5          THEN
;  1261          1321  6              BEGIN
;  1262          1322  6              IRAB [IRB$L_VBN_LEFT] = 0;
;  1263          1323  6              IRAB [IRB$L_VBN_RIGHT] = 0;
;  1264          1324  7              RETURN RMSSOC(SOC)
;  1265          1325  5              END;
;  1266          1326  5
;  1267          1327  5          IF .ST LSS 0
;  1268          1328  5              AND
;  1269          1329  6              .REC_ADDR EQLA (.BKT_ADDR + BKT$C_OVERHDSZ)
;  1270          1330  5              OR
;  1271          1331  6              (.BKT_ADDR[BKT$V_LASTBKT] AND .IRAB[IRB$V_EMPT_SEEN])
;  1272          1332  5          THEN
;  1273          1333  6              BEGIN
;  1274          1334  6
;  1275          1335  6              IF .IRAB [IRB$L_VBN_RIGHT] EQL 0
;  1276          1336  6              THEN
;  1277          1337  6                  IRAB [IRB$L_VBN_RIGHT] = .BDB [BDB$L_VBN];
;  1278          1338  6
;  1279          1339  6              VBN = .IRAB [IRB$L_VBN_LEFT];
;  1280          1340  6              END
;  1281          1341  6
;  1282          1342  5          ELSE
;  1283          1343  6              BEGIN
;  1284          1344  6
;  1285          1345  6              IF .ST LEQ 0 OR .BKT_ADDR[BKT$V_LASTBKT]
;  1286          1346  6              THEN
;  1287          1347  7                  BEGIN
```

```
;  1288     1348  7                IRAB [IRB$L_VBN_LEFT] = 0;
;  1289     1349  7                IRAB [IRB$L_VBN_RIGHT] = 0;
;  1290     1350  7                RETURN RMSSOC(SOC);
;  1291     1351  6                END;
;  1292     1352  6
;  1293     1353  6            IF .IRAB [IRB$V_EMPT_SEEN]
;  1294     1354  6            THEN
;  1295     1355  7                BEGIN
;  1296     1356  7                IF .IRAB [IRB$L_VBN_RIGHT] EQL 0
;  1297     1357  7                THEN
;  1298     1358  7                    IRAB [IRB$L_VBN_RIGHT] = .BDB [BDB$L_VBN];
;  1299     1359  7                END
;  1300     1360  6            ELSE
;  1301     1361  7                BEGIN
;  1302     1362  7                IRAB [IRB$L_VBN_RIGHT] = 0;
;  1303     1363  7                IRAB [IRB$L_VBN_LEFT] = .BDB [BDB$L_VBN];
;  1304     1364  6                END;
;  1305     1365  5            END;
;  1306     1366  5
;  1307     1367  5        RM$RLSBKT(0);
;  1308     1368  5        IRAB [IRB$L_CURBDB] = 0;
;  1309     1369  5        IRAB [IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  1310     1370  5
;  1311     1371  6        IF NOT (ST=RM$GETBKT(.VBN,.SIZE))
;  1312     1372  5        THEN
;  1313     1373  6            BEGIN
;  1314     1374  6            IF TESTBITSC (IRAB[IRB$V_ABOVELCKD])
;  1315     1375  6            THEN
;  1316     1376  6                RELEASE(IRAB[IRB$L_LOCK_BDB]);
;  1317     1377  6            RETURN .ST
;  1318     1378  5            END;
;  1319     1379  5
;  1320     1380  5        IRAB [IRB$L_CURBDB] = .BDB;
;  1321     1381  5
;  1322     1382  4            END;    ! of position for insert level 0 WHILE loop.
;  1323     1383  4        END         ! of greater than status from srch_by_key.
;  1324     1384  3    ELSE
;  1325     1385  3
;  1326     1386  3        ! status was less than or equal meaning we have a down pointer if
;  1327     1387  3        ! at an index level or may or may not have the data record
;  1328     1388  3        ! depending upon whether an exact match was desired or not
;  1329     1389  3        !
;  1330     1390  4        BEGIN
;  1331     1391  4
;  1332     1392  4        LOCAL
;  1333     1393  4            SIZE;
;  1334     1394  4
;  1335     1395  4        IF .BKT_ADDR[BKT$B_LEVEL] EQLU .IRAB[IRB$B_STOPLEVEL]
;  1336     1396  4        THEN
;  1337     1397  5            BEGIN
;  1338     1398  5
;  1339     1399  5            IF .ST EQL 0
;  1340     1400  5            THEN
;  1341     1401  6                RETURN RMSSUC(SUC)
;  1342     1402  5            ELSE
;  1343     1403  5
;  1344     1404  5                IF .IRAB[IRB$V_POSINSERT]
```

```
;  1345     1405  5                    OR
;  1346     1406  5                    .IRAB[IRB$V_SRCHGT]
;  1347     1407  5                    OR
;  1348     1408  5                    .IRAB[IRB$V_SRCHGE]
;  1349     1409  5                THEN
;  1350     1410  6                    RETURN RMSSUC(SUC)
;  1351     1411  5                ELSE
;  1352     1412  5                    RETURN RMSERR(RNF);
;  1353     1413  5
;  1354     1414  4            END;
;  1355     1415  4
;  1356     1416  4            ! Get the down pointer from the index record positioned at.
;  1357     1417  4            !
;  1358     1418  4
;  1359     1419  4            IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  1360     1420  4            THEN
;  1361     1421  5                BEGIN
;  1362     1422  5                AP = 1;
;  1363     1423  5                VBN = RM$RFA_ORD_VBN();
;  1364     1424  5                END
;  1365     1425  4            ELSE            ! Version 3 index bucket
;  1366     1426  5                BEGIN
;  1367     1427  5                VBN = RM$V3_VBN();
;  1368     1428  4                END;
;  1369     1429  4
;  1370     1430  4            IF .VBN EQLU 0
;  1371     1431  4            THEN
;  1372     1432  4                RETURN (RMSERR(TRE));
;  1373     1433  4
;  1374     1434  4            ! Check if about to read data level and switch to data level
;  1375     1435  4            ! transfer size.
;  1376     1436  4            !
;  1377     1437  4            IF .BKT_ADDR[BKT$B_LEVEL] EQLU 1
;  1378     1438  4            THEN
;  1379     1439  4                SIZE = .IDX_DFN[IDX$B_DATBKTSZ]
;  1380     1440  4            ELSE
;  1381     1441  4                SIZE = .IDX_DFN[IDX$B_IDXBKTSZ];
;  1382     1442  4
;  1383     1443  4            SIZE = .SIZE*512;
;  1384     1444  4
;  1385     1445  4            ! now check to see if the next level to read needs to be locked
;  1386     1446  4            !
;  1387     1447  4            !
;  1388     1448  4            BDB = .IRAB[IRB$L_CURBDB];          ! get curbdb into bdb
;  1389     1449  4
;  1390     1450  4            IF .IRAB[IRB$V_POSINSERT]
;  1391     1451  4                OR
;  1392     1452  4                .IRAB[IRB$V_POSDELETE]
;  1393     1453  4            THEN
;  1394     1454  4
;  1395     1455  4                ! lock the stop level if either position for insert or delete
;  1396     1456  4                !
;  1397     1457  4                !
;  1398     1458  4 BLK4 :          BEGIN
;  1399     1459  5
;  1400     1460  5                LOCAL
;  1401     1461  5
```

```
;  1402          1462  5                  STOPLEVCHK  : BYTE;
;  1403          1463  5
;  1404          1464  5                STOPLEVCHK = .IRAB[IRB$B_STOPLEVEL];
;  1405          1465  5                STOPLEVCHK = .STOPLEVCHK + 1;
;  1406          1466  5
;  1407          1467  5                IF .BKT_ADDR[BKT$B_LEVEL] EQL .STOPLEVCHK
;  1408          1468  5                THEN
;  1409          1469  5 BLK5 :
;  1410          1470  6                    BEGIN
;  1411          1471  6
;  1412          1472  6                    LOCAL
;  1413          1473  6                        REC_SIZE;
;  1414          1474  6
;  1415          1475  6                    ! if the stoplevel not zero, just lock it
;  1416          1476  6                    !
;  1417          1477  6
;  1418          1478  6                    IF .IRAB[IRB$B_STOPLEVEL] NEQ 0
;  1419          1479  6                        OR
;  1420          1480  6                        .IRAB[IRB$V_POSDELETE]
;  1421          1481  6                    THEN
;  1422          1482  7                        BEGIN
;  1423          1483  7                        IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  1424          1484  7                        LEAVE BLK5;
;  1425          1485  7
;  1426          1486  6                        END;
;  1427          1487  6
;  1428          1488  6                    ! try to save the next down pointer if it's in this bucket
;  1429          1489  6                    ! to avoid possible i/o at the data level
;  1430          1490  6                    !
;  1431          1491  7 BLK6 :              BEGIN
;  1432          1492  7
;  1433          1493  7                    IRAB [IRB$L_NEXT_DOWN] = -1;
;  1434          1494  7
;  1435          1495  7                    ! See if we're already at the end of the bucket.
;  1436          1496  7                    !
;  1437          1497  7                    CASE .IDX_DFN [IDX$B_IDXBKTYP] FROM IDX$C_V2_BKT TO IDX$C_NCMPIDX OF
;  1438          1498  7                    SET
;  1439          1499  8                    [IDX$C_V2_BKT]: BEGIN
;  1440          1500  8                                    REC_ADDR = .REC_ADDR + RM$REC_OVHD(1; REC_SIZE);
;  1441          1501  8                                    REC_ADDR = .REC_ADDR + .REC_SIZE;
;  1442          1502  7                                    END;
;  1443          1503  7
;  1444          1504  8                    [IDX$C_CMPIDX]: BEGIN
;  1445          1505  8                                    REC_ADDR = .REC_ADDR + .(.REC_ADDR)<0,8> + 2;
;  1446          1506  7                                    END;
;  1447          1507  7
;  1448          1508  8                    [IDX$C_NCMPIDX]:BEGIN
;  1449          1509  8                                    REC_ADDR = .REC_ADDR + .IDX_DFN [IDX$B_KEYSZ];
;  1450          1510  7                                    END;
;  1451          1511  7                    TES;
;  1452          1512  7
;  1453          1513  7                    IF .REC_ADDR GEQA .BKT_ADDR + .BKT_ADDR[BKT$W_FREESPACE]
;  1454          1514  7                    THEN
;  1455          1515  7                        LEAVE BLK6;
;  1456          1516  7
;  1457          1517  7                    IF .IDX_DFN [IDX$B_IDXBKTYP] EQL IDX$C_V2_BKT
;  1458          1518  7                    THEN
```

```
;  1459          1519  8                    BEGIN
;  1460          1520  8                    AP = 1;
;  1461          1521  8                    IRAB [IRB$L_NEXT_DOWN] = RM$RECORD_VBN();
;  1462          1522  8                    END
;  1463          1523  7                ELSE
;  1464          1524  8                    BEGIN
;  1465          1525  8                    IRAB [IRB$L_REC_COUNT] = .IRAB [IRB$L_REC_COUNT] + 1;
;  1466          1526  8                    IRAB [IRB$L_NEXT_DOWN] = RM$V3_VBN();
;  1467          1527  7                    END;
;  1468          1528  7
;  1469          1529  6                END;  ! of BLK6
;  1470          1530  6
;  1471          1531  6                ! if we have extrabufs and this bucket is locked then
;  1472          1532  6                ! we did lockabove optimization so set
;  1473          1533  6                ! abovelckd to note that and save the bdb in lock_bdb.
;  1474          1534  6                ! Also, if the blb_ptr is zero and there are extrabufs
;  1475          1535  6                ! then the file is exclusively accessed therefore the
;  1476          1536  6                ! optimization may also be made.
;  1477          1537  6                !
;  1478          1538  6
;  1479          1539  6                IF  .IRAB[IRB$B_BCNT] GTRU 2
;  1480          1540  6                THEN
;  1481          1541  6                    IF .BDB[BDB$L_BLB_PTR] EQL 0
;  1482          1542  6                    THEN
;  1483          1543  7                        BEGIN
;  1484          1544  7                        IRAB[IRB$V_ABOVELCKD] = 1;
;  1485          1545  7                        IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK OR CSH$M_NOWAIT;
;  1486          1546  7                        IRAB[IRB$L_LOCK_BDB] = .BDB;
;  1487          1547  7                        IRAB[IRB$L_CURBDB] = 0;
;  1488          1548  7                        LEAVE BLK4;
;  1489          1549  7
;  1490          1550  7                        END
;  1491          1551  6                    ELSE
;  1492          1552  6                        IF .BBLOCK[.BDB[BDB$L_BLB_PTR], BLB$V_LOCK]
;  1493          1553  6                        THEN
;  1494          1554  7                            BEGIN
;  1495          1555  7                            IRAB[IRB$V_ABOVELCKD] = 1;
;  1496          1556  7                            IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK OR CSH$M_NOWAIT;
;  1497          1557  7                            IRAB[IRB$L_LOCK_BDB] = .BDB;
;  1498          1558  7                            IRAB[IRB$L_CURBDB] = 0;
;  1499          1559  7                            LEAVE BLK4;
;  1500          1560  7
;  1501          1561  6                            END;
;  1502          1562  6
;  1503          1563  6                    ! We are not keeping the level above locked on the
;  1504          1564  6                    ! way down.  Remember the vbn of this level 1 bucket we
;  1505          1565  6                    ! just passed through because it may have to be
;  1506          1566  6                    ! checked when we get to level 0.
;  1507          1567  6                    !
;  1508          1568  6
;  1509          1569  6                    IRAB [IRB$L_VBN_RIGHT] = .BDB [BDB$L_VBN];
;  1510          1570  6                    IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  1511          1571  6
;  1512          1572  6                    END
;  1513          1573  5                ELSE
;  1514          1574  5
;  1515          1575  5                    ! if position for insert and we have the extra buffers and
```

```
;1516     1576  5                ! we are going all the way to level zero, then we will try
;1517     1577  5                ! the performance optimization of keeping the level above
;1518     1578  5                ! the data level locked to avoid going back to the root in
;1519     1579  5                ! the event of a split.
;1520     1580  5                !
;1521     1581  5
;1522     1582  5                IF .IRAB[IRB$V_POSINSERT]
;1523     1583  5                    AND
;1524     1584  5                    .IRAB[IRB$B_STOPLEVEL] EQL 0
;1525     1585  5                    AND
;1526     1586  5                    .IRAB[IRB$B_BCNT] GTR 2
;1527     1587  5                THEN
;1528     1588  6                    BEGIN
;1529     1589  6                    STOPLEVCHK = .STOPLEVCHK + 1;
;1530     1590  6
;1531     1591  6                    IF .BKT_ADDR[BKT$B_LEVEL] EQL .STOPLEVCHK
;1532     1592  6                    THEN
;1533     1593  6                        BBLOCK[IRAB[IRB$B_CACHEFLGS], CSH$V_LOCK] = 1;
;1534     1594  6
;1535     1595  5                    END;
;1536     1596  5
;1537     1597  5                ! release the current bucket before going down
;1538     1598  5                !
;1539     1599  5                RM$RLSBKT(0);
;1540     1600  5                IRAB[IRB$L_CURBDB] = 0;
;1541     1601  5                END                                 ! of BLK4
;1542     1602  5
;1543     1603  5            ! This is not a position for modification. Release the level above
;1544     1604  5            ! bucket, and if the next bucket to be obtained is a data bucket,
;1545     1605  5            ! the file is write accessed, and either the file is RU Journallable
;1546     1606  5            ! or the data bucket to be obtained is a SIDR bucket, then make
;1547     1607  5            ! sure the next bucket is locked.
;1548     1608  5            !
;1549     1609  4            ELSE
;1550     1610  5                BEGIN
;1551     1611  5
;1552     1612  5                ! Determine whether the next bucket to be obtained will have to
;1553     1613  5                ! be locked.
;1554     1614  5                !
;1555     1615  5                IF .IFAB[IFB$V_WRTACC]
;1556     1616  5                    AND
;1557     1617  5                   .BKT_ADDR[BKT$B_LEVEL] EQLU 1
;1558     1618  5                    AND
;1559     1619  6                   (.IFAB[IFB$V_RU]
;1560     1620  6                         OR
;1561     1621  6                        .IDX_DFN[IDX$B_KEYREF] GTRU 0)
;1562     1622  5                THEN
;1563     1623  5                    IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;1564     1624  5
;1565     1625  5                ! Release the current bucket.
;1566     1626  5                !
;1567     1627  5                RM$RLSBKT(0);
;1568     1628  5                IRAB[IRB$L_CURBDB] = 0;
;1569     1629  4                END;
;1570     1630  4
;1571     1631  4            WHILE 1 DO
;1572     1632  5                BEGIN
```

```
;  1573          1633  5              ST = RM$GETBKT(.VBN, .SIZE);
;  1574          1634  5
;  1575          1635  5              ! if successful drop out of here
;  1576          1636  5              !
;  1577          1637  5
;  1578          1638  5              IF .ST
;  1579          1639  5              THEN
;  1580          1640  5                  EXITLOOP;
;  1581          1641  5
;  1582          1642  5              ! if abovelckd then release it to
;  1583          1643  5              ! avoid deadlock.  Save the VBN of the level 1
;  1584          1644  5              ! bucket as it will need to be rechecked when level 0
;  1585          1645  5              ! is reached.
;  1586          1646  5              !
;  1587          1647  5
;  1588          1648  5              IF TESTBITSC(IRAB[IRB$V_ABOVELCKD])
;  1589          1649  5              THEN
;  1590          1650  6                  BEGIN
;  1591          1651  6                  BDB = .IRAB[IRB$L_LOCK_BDB];
;  1592          1652  6                  IRAB [IRB$L_VBN_RIGHT] = .BDB [BDB$L_VBN];
;  1593          1653  6                  IRAB[IRB$L_LOCK_BDB] = 0;
;  1594          1654  6                  RM$RLSBKT(0);
;  1595          1655  5                  END;
;  1596          1656  5
;  1597          1657  6              IF .ST<0,16> NEQ RMSERR(RLK)
;  1598          1658  5              THEN RETURN .ST;
;  1599          1659  5
;  1600          1660  5              IRAB [IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  1601          1661  4              END;
;  1602          1662  4
;  1603          1663  4          IRAB[IRB$L_CURBDB] = .BDB;
;  1604          1664  4              VBN = 0;
;  1605          1665  3              END;                                    ! end else block
;  1606          1666  3
;  1607          1667  3          REC_ADDR = .BKT_ADDR + BKT$C_OVERHDSZ;
;  1608          1668  3          IRAB [IRB$L_REC_COUNT] = 0;
;  1609          1669  3          END
;  1610          1670  2      UNTIL 0                                         ! end until loop
;  1611          1671  1      END;


                                      0C  BB 000C0 RM$SEARCH_TREE:
                                                   POSHR   #^M<R2,R3>                          ; 0892
                          5E          0C  C2 00002         SUBL2   #12, SP                     ;
                    04    AE          D4  D4 00005         CLRL    VBN                         ; 0984
                          6E          D4  D4 00008 1$:     CLRL    ST                          ; 0993
                07        42    A9    06  E0 0000A         BBS     #6, 66(IRAB), 2$            ; 0998
                                      84  10 0000F         BSBB    RM$SRCH_BY_KEY              ; 1000
                          6E          50  D0 00011         MOVL    R0, ST                      ;
                                      06  11 00014         BRB     3$                          ;
                      0098    C9    0E  A5  9E 00016 2$:   MOVAB   14(R5), 152(IRAB)           ; 1002
                                      6E  D5 0001C 3$:     TSTL    ST                          ; 1010
                                      03  14 0001E         BGTR    4$                          ;
                                    0201  31 00020         BRW     39$                         ;
```

```
                07      0D   A5  E9 00023 4$:     BLBC    13(BKT_ADDR), 5$                        ;1017
                31      42   A9  E8 00027         BLBS    66(IRAB), 9$                            ;1020
                          0214  31  0002B         BRW     41$
                54      20   A9  D0 0002E 5$:     MOVL    32(IRAB), BDB                           ;1026
          08    AE      14   A4  3C 00032         MOVZWL  20(BDB), SIZE                           ;1027
                        04   AE  D5 00037         TSTL    VBN                                     ;1029
                        05   12  0003A           BNEQ    6$
          04    AE      08   A5  D0 0003C         MOVL    8(BKT_ADDR), VBN                        ;1031
                03      42   A9  E8 00041 6$:     BLBS    66(IRAB), 8$                            ;1033
                          00D0  31  00045 7$:     BRW     20$
                        0C   A5  95 00048 8$:     TSTB    12(BKT_ADDR)
                        F8   12  0004B           BNEQ    7$
                50    0090   C9  D0 0004D         MOVL    144(IRAB), R0                           ;1048
                50      04   AE  D1 00052         CMPL    VBN, R0
                        0C   12  00056           BNEQ    11$
                03      1C   A7  E8 00058         BLBS    28(IDX_DFN), 10$                        ;1053
                          01DE  31  0005C 9$:     BRW     40$
                        21   A7  95 0005F 10$:    TSTB    33(IDX_DFN)
                        F8   12  00062           BNEQ    9$
    05    06    AA          03  E0 00064 11$:    BBS     #3, 6(IFAB), 12$                        ;1094
    07    06    A9          05  E1 00069         BBC     #5, 6(IRAB), 13$                        ;1095
                          50   D5 0006E 12$:     TSTL    R0                                      ;1098
                          D3   13  00070         BEQL    7$
                        0099  31  00072         BRW     19$                                     ;1101
                          50   D5 00075 13$:     TSTL    R0                                      ;1114
                          CC   13  00077         BEQL    7$
                52      16   A7  9A 00079         MOVZBL  22(IDX_DFN), R2                         ;1125
    52          52          09  78 0007D         ASHL    #9, R2, R2
                          53   D4  00081         CLRL    R3
                51    008C   C9  D0 00083         MOVL    140(IRAB), R1                           ;1126
                00000000G   00  16 00088         JSB     RM$CACHE
                6E          50  D0 0008E         MOVL    R0, ST
                1C          6E  E9 00091         BLBC    ST, 14$                                 ;1142
                56        0E   A5  9E 00094         MOVAB   14(R5), REC_ADDR                        ;1143
                        0094   C9  D4 00098         CLRL    148(IRAB)                               ;1144
                             FEF6  30  0009C         BSBW    RM$SRCH_BY_KEY
                          50   D5  0009F         TSTL    R0                                      ;1147
                          1F   15  000A1         BLEQ    15$                                     ;1148
                6E      82AA   8F  3C 000A3         MOVZWL  #33450, ST
                          7E   D4  000A8         CLRL    -(SP)
                        0000G  30  000AA         BSBW    RM$RLSBKT
                5E          04  C0  000AD         ADDL2   #4, SP
                54        20   A9  D0 000B0 14$:     MOVL    32(IRAB), BDB                           ;1149
                          7E   D4  000B4         CLRL    -(SP)                                   ;1150
                        0000G  30  000B6         BSBW    RM$RLSBKT
                5E          04  C0  000B9         ADDL2   #4, SP
                          20   A9  D4  000BC         CLRL    32(IRAB)                                ;1151
                          02D6  31  000BF         BRW     69$                                     ;1152
                52        20   A9  D0 000C2 15$:     MOVL    32(IRAB), LEV0_BDB                      ;1166
                5C          01  D0  000C6         MOVL    #1, AP                                  ;1167
                03      00B7   CA  91 000C9         CMPB    183(IFAB), #3                           ;1169
                          05   1E  000CE         BGEQU   16$
                        0000G  30  000D0         BSBW    RM$RECORD_VBN                           ;1171
                          03   11  000D3         BRB     17$
                        0000V  30  000D5 16$:     BSBW    RM$V3_VBN                               ;1173
          1C    A2          50  D1  000D8 17$:     CMPL    LEV1_VBN, 28(LEV0_BDB)                  ;1175
                          21   13  000DC         BEQL    18$
```

```
                        44   A9  94 000DE          CLRB    68(IRAB)                                ; 1184
                    20  A9       54 DO 000E1       MOVL    BDB, 32(IRAB)                           ; 1185
                    54           52 DO 000E5       MOVL    LEVO_BDB, BDB                           ; 1186
                                 7E D4 000E8       CLRL    -(SP)                                   ; 1187
                              0000G 30 000EA       BSBW    RM$RLSBKT                               ;
                        5E         04 C0 000ED      ADDL2   #4, SP                                  ;
                        54   20    A9 DO 000F0      MOVL    32(IRAB), BDB                           ; 1188
                        55   18    A4 DO 000F4      MOVL    24(BDB), BKT_ADDR                       ; 1189
                        56   0E    A5 9E 000F8      MOVAB   14(R5), REC_ADDR                        ; 1190
                           FF09    31 000FC         BRW     1$                                      ; 1191
                                 7E D4 000FF 18$:   CLRL    -(SP)                                   ; 1199
                              0000G 30 00101       BSBW    RM$RLSBKT                               ;
                        5E         04 C0 00104      ADDL2   #4, SP                                  ;
                        54         52 DO 00107      MOVL    LEVO_BDB, BDB                           ; 1200
                        55   18    A4 DO 0010A      MOVL    24(BDB), BKT_ADDR                       ; 1201
                           008C    C9 7C 0010E 19$:   CLRQ    140(IRAB)                               ; 1204
                    0088   C9   1C A4 DO 00112      MOVL    28(BDB), 136(IRAB)                      ; 1206
                        05   42    A9 93 00118 20$:   BITB    66(IRAB), #5                            ; 1220
                                   07 13 0011C      BEQL    21$                                     ;
                    0C  A5       41 A9 91 0011E    CMPB    65(IRAB), 12(BKT_ADDR)                  ; 1222
                                   1A 13 00123      BEQL    22$                                     ;
                        05   42    A9 93 00125 21$:   BITB    66(IRAB), #5                            ; 1226
                                   18 12 00129      BNEQ    23$                                     ;
                        14   06    AA E9 0012B      BLBC    6(IFAB), 23$                            ; 1228
                             0C    A5 95 0012F      TSTB    12(BKT_ADDR)                            ; 1230
                                   0F 12 00132      BNEQ    23$                                     ;
                05      00A0   CA    01 E0 00134      BBS     #1, 160(IFAB), 22$                      ; 1232
                             21    A7 95 0013A      TSTB    33(IDX_DFN)                             ; 1234
                                   04 13 0013D      BEQL    23$                                     ;
                        40   A9       01 90 0013F 22$:   MOVB    #1, 64(IRAB)                            ; 1236
                                 7E D4 00143 23$:   CLRL    -(SP)                                   ; 1241
                              0000G 30 00145       BSBW    RM$RLSBKT                               ;
                        5E         04 C0 00148      ADDL2   #4, SP                                  ;
                             20    A9 D4 0014B      CLRL    32(IRAB)                                ; 1242
                             08    AE DD 0014E      PUSHL   SIZE                                    ; 1247
                             08    AE DD 00151      PUSHL   VBN                                     ;
                                0000G 30 00154      BSBW    RM$GETBKT                               ;
                    5E             08 C0 00157      ADDL2   #8, SP                                  ;
                    6E             50 DO 0015A      MOVL    R0, ST                                  ;
                    03             6E E8 0015D      BLBS    ST, 24$                                 ;
                                00A1 31 00160       BRW     36$                                     ;
                04  AE         08   A5 DO 00163 24$:   MOVL    8(BKT_ADDR), VBN                        ; 1261
                20  A9             54 DO 00168      MOVL    BDB, 32(IRAB)                           ; 1262
                    03         42   A9 E8 0016C      BLBS    66(IRAB), 26$                           ; 1264
                                0237 31 00170 25$:   BRW     72$                                     ;
                             0C    A5 95 00173 26$:   TSTB    12(BKT_ADDR)                            ;
                                   F8 12 00176      BNEQ    25$                                     ;
                        56         0E   A5 9E 00178 27$:   MOVAB   14(R5), REC_ADDR                 ; 1314
                           0094    C9   D4 0017C      CLRL    148(IRAB)                               ; 1315
                                FE12    30 00180      BSBW    RM$SRCH_BY_KEY                          ; 1317
                    6E             50 DO 00183      MOVL    R0, ST                                  ;
                04  AE         08   A5 DO 00186      MOVL    8(BKT_ADDR), VBN                        ; 1319
                    50         008C C9 9E 0018B      MOVAB   140(IRAB), R0                           ;
                    60         04   AE D1 00190      CMPL    VBN, (R0)                               ;
                                   2E 13 00194      BEQL    32$                                     ;
                                   6E D5 00196      TSTL    ST                                      ; 1327
                                   09 18 00198      BGEQ    28$                                     ;
```

```
                51      0E      A5  9E 0019A          MOVAB   14(R5), R1                      ; 1329
                51              56  D1 0019E          CMPL    REC_ADDR, R1                    ;
                                09  13 001A1          BEQL    29$                             ;
                15      0D      A5  E9 001A3 28$:     BLBC    13(BKT_ADDR), 31$               ; 1331
        10      44      A9          01  E1 001A7      BBC     #1, 68(IRAB), 31$               ;
                                    60  D5 001AC 29$:  TSTL    (R0)                            ; 1335
                                    04  12 001AE      BNEQ    30$                             ;
                60      1C      A4  D0 001B0          MOVL    28(BDB), (R0)                   ; 1337
                04      AE  0088    C9  D0 001B4 30$:  MOVL    136(IRAB), VBN                  ; 1339
                                    27  11 001BA      BRB     35$                             ; 1327
                                    6E  D5 001BC 31$:  TSTL    ST                              ; 1345
                                    04  15 001BE      BLEQ    32$                             ;
                08      0D      A5  E9 001C0          BLBC    13(BKT_ADDR), 33$               ;
                    0088        C9  D4 001C4 32$:     CLRL    136(IRAB)                       ; 1348
                                    60  D4 001C8      CLRL    (R0)                            ; 1349
                                    71  11 001CA      BRB     40$                             ; 1350
        0A      44      A9          01  E1 001CC 33$:  BBC     #1, 68(IRAB), 34$               ; 1353
                                    60  D5 001D1      TSTL    (R0)                            ; 1356
                                    0E  12 001D3      BNEQ    35$                             ;
                60      1C      A4  D0 001D5          MOVL    28(BDB), (R0)                   ; 1358
                                    08  11 001D9      BRB     35$                             ; 1353
                                    60  D4 001DB 34$:  CLRL    (R0)                            ; 1362
            0088    C9      1C      A4  D0 001DD      MOVL    28(BDB), 136(IRAB)              ; 1363
                                    7E  D4 001E3 35$:  CLRL    -(SP)                           ; 1367
                                0000G   30 001E5      BSBW    RM$RLSBKT                       ;
                5E                  04  C0 001E8      ADDL2   #4, SP                          ;
                    20      A9      D4 001EB          CLRL    32(IRAB)                        ; 1368
                40      A9          01  90 001EE      MOVB    #1, 64(IRAB)                    ; 1369
                                08  AE  DD 001F2      PUSHL   SIZE                            ; 1371
                                08  AE  DD 001F5      PUSHL   VBN                             ;
                                0000G   30 001F8      BSBW    RM$GETBKT                       ;
                5E                  08  C0 001FB      ADDL2   #8, SP                          ;
                6E                  50  D0 001FE      MOVL    R0, ST                          ;
                19                  6E  E8 00201      BLBS    ST, 38$                         ;
        11      04      A9          15  E5 00204 36$:  BBCC    #21, 4(IRAB), 37$               ; 1374
                54      0084        C9  D0 00209      MOVL    132(IRAB), BDB                  ; 1376
                            0084    C9  D4 0020E      CLRL    132(IRAB)                       ;
                                    7E  D4 00212      CLRL    -(SP)                           ;
                                0000G   30 00214      BSBW    RM$RLSBKT                       ;
                5E                  04  C0 00217      ADDL2   #4, SP                          ;
                                017B    31 0021A 37$:  BRW     69$                             ; 1377
                    20      A9      54  D0 0021D 38$:  MOVL    BDB, 32(IRAB)                   ; 1380
                                FF54    31 00221      BRW     27$                             ; 1310
                    41      A9  0C      A5  91 00224 39$:  CMPB    12(BKT_ADDR), 65(IRAB)          ; 1395
                                    1E  12 00229      BNEQ    42$                             ;
                                    6E  D5 0022B      TSTL    ST                              ; 1399
                                    0E  13 0022D      BEQL    40$                             ;
                0A          42      A9  E8 0022F      BLBS    66(IRAB), 40$                   ; 1404
        05      42      A9          01  E0 00233      BBS     #1, 66(IRAB), 40$               ; 1406
        05      42      A9          04  E1 00238      BBC     #4, 66(IRAB), 41$               ; 1408
                50                  01  D0 0023D 40$:  MOVL    #1, R0                          ; 1410
                                    24  11 00240      BRB     45$                             ;
                50      82B2        8F  3C 00242 41$:  MOVZWL  #33458, R0                      ; 1412
                                    1D  11 00247      BRB     45$                             ; 1404
                03      00B7        CA  91 00249 42$:  CMPB    183(IFAB), #3                   ; 1419
                                    08  1E 0024E      BGEQU   43$                             ;
                5C                  01  D0 00250      MOVL    #1, AP                          ; 1422
```

```
                                0000G 30 00253          BSBW    RM$RECORD_VBN                                   ; 1423
                                   03 11 00256          BRB     44$
                                0000v 30 00258 43$:     BSBW    RM$V3_VBN                                       ; 1427
                          04    AE    50 D0 0025B 44$:  MOVL    R0, VBN                                         ;
                                      08 12 0025F       BNEQ    46$                                             ; 1430
                                50    86DC 8F 3C 00261  MOVZWL  #34524, R0                                      ; 1432
                                   014C 31 00266 45$:   BRW     73$
                                      51 D4 00269 46$:  CLRL    R1                                              ; 1437
                                01    0C   A5 91 0026B  CMPB    12(BKT_ADDR), #1                                ;
                                      08 12 0026F       BNEQ    47$                                             ;
                                      51 D6 00271       INCL    R1                                              ;
                                53    17   A7 9A 00273  MOVZBL  23(IDX_DFN), SIZE                              ; 1439
                                      04 11 00277       BRB     48$                                             ;
                                53    16   A7 9A 00279 47$:  MOVZBL  22(IDX_DFN), SIZE                         ; 1441
                          53          53   09 78 0027D 48$:  ASHL    #9, SIZE, SIZE                              ; 1443
                                54    20   A9 D0 00281       MOVL    32(IRAB), BDB                               ; 1448
                                08    42   A9 E8 00285       BLBS    66(IRAB), 49$                               ; 1450
                          03    42    A9    02 E0 00289       BBS     #2, 66(IRAB), 49$                           ; 1452
                                   00B2 31 0028E       BRW     63$
                                50    41   A9 90 00291 49$:  MOVB    65(IRAB), STOPLEVCHK                        ; 1464
                                      50 96 00295       INCB    STOPLEVCHK                                      ; 1465
                                50    0C   A5 91 00297  CMPB    12(BKT_ADDR), STOPLEVCHK                        ; 1467
                                      03 13 0029B       BEQL    50$                                             ;
                                   0086 31 0029D       BRW     62$                                             ;
                                41    A9 95 002A0 50$:  TSTB    65(IRAB)                                        ; 1478
                                      7F 12 002A3       BNEQ    61$                                             ;
                          7A    42    A9    02 E0 002A5       BBS     #2, 66(IRAB), 61$                           ; 1480
                                52    0090 C9 9E 002AA  MOVAB   144(IRAB), R2                                   ; 1493
                                62       01 CE 002AF    MNEGL   #1, (R2)                                        ;
                          02          00    28 A7 8F 002B2    CASEB   40(IDX_DFN), #0, #2                       ; 1497
                        001E        0014      0006      002B7 51$:  .WORD   52$-51$,-                             ;
                                                                          53$-51$,-                             ;
                                                                          54$-51$                               ;
                                51          01 D0 002BD 52$:  MOVL    #1, R1                                  ; 1500
                                         0000G 30 002C0       BSBW    RM$REC_OVHD                             ;
                                56          50 C0 002C3       ADDL2   R0, REC_ADDR                            ;
                                56          51 C0 002C6       ADDL2   REC_SIZE, REC_ADDR                      ; 1501
                                            11 11 002C9       BRB     55$                                     ; 1497
                                50          66 9A 002CB 53$:  MOVZBL  (REC_ADDR), R0                          ; 1505
                                56       02 A046 9E 002CE     MOVAB   2(R0)[REC_ADDR], REC_ADDR               ;
                                            07 11 002D3       BRB     55$                                     ; 1497
                                50       20   A7 9A 002D5 54$:  MOVZBL  32(IDX_DFN), R0                      ; 1509
                                56          50 C0 002D9       ADDL2   R0, REC_ADDR                            ;
                                50       04   A5 3C 002DC 55$:  MOVZWL  4(BKT_ADDR), R0                      ; 1513
                                50          55 C0 002E0       ADDL2   BKT_ADDR, R0                            ;
                                50          56 D1 002E3       CMPL    REC_ADDR, R0                            ;
                                            17 1E 002E6       BGEQU   58$                                     ;
                                         28   A7 95 002E8       TSTB    40(IDX_DFN)                             ; 1517
                                            08 12 002EB       BNEQ    56$                                     ;
                                5C          01 D0 002ED       MOVL    #1, AP                                  ; 1520
                                         0000G 30 002F0       BSBW    RM$RECORD_VBN                           ; 1521
                                            07 11 002F3       BRB     57$                                     ;
                                      0094   C9 D6 002F5 56$:  INCL    148(IRAB)                               ; 1525
                                         0000V 30 002F9       BSBW    RM$V3_VBN                               ; 1526
                                62          50 D0 002FC 57$:  MOVL    R0, (R2)                                ;
                                02       54   A9 91 002FF 58$:  CMPB    84(IRAB), #2                            ; 1539
                                            19 1B 00303       BLEQU   60$                                     ;
```

```
                50        10    A4  D0 00305          MOVL    16(BDB), R0                          ; 1541
                          04    13  00309          BEQL    59$
                0F        0A    A0  E9 0030B          BLBC    10(R0), 60$                          ; 1552
              06  A9            20  88 0030F 59$:     BISB2   #32, 6(IRAB)                         ; 1555
              40  A9            03  90 00313          MOVB    #3, 64(IRAB)                         ; 1556
            0084  C9            54  D0 00317          MOVL    BDB, 132(IRAB)                       ; 1557
                                43  11 0031C          BRB     66$                                  ; 1558
            008C  C9        1C  A4  D0 0031E 60$:     MOVL    28(BDB), 140(IRAB)                   ; 1569
                                2F  11 00324 61$:     BRB     64$                                  ; 1570
                2F        42  A9  E9 00326 62$:     BLBC    66(IRAB), 65$                        ; 1582
                          41  A9  95 0032A          TSTB    65(IRAB)                             ; 1584
                              2A  12 0032D          BNEQ    65$
                02        54  A9  91 0032F          CMPB    84(IRAB), #2                         ; 1586
                              24  1B 00333          BLEQU   65$
                              50  96 00335          INCB    STOPLEVCHK                           ; 1589
                50        0C  A5  91 00337          CMPB    12(BKT_ADDR), STOPLEVCHK             ; 1591
                              1C  12 0033B          BNEQ    65$
              40  A9            01  88 0033D          BISB2   #1, 64(IRAB)                         ; 1593
                                16  11 00341          BRB     65$                                  ; 1599
                12        06  AA  E9 00343 63$:     BLBC    6(IFAB), 65$                         ; 1615
                0F              51  E9 00347          BLBC    R1, 65$                              ; 1617
    05        00A0  CA            01  E0 0034A          BBS     #1, 160(IFAB), 64$                   ; 1619
                        21  A7  95 00350          TSTB    33(IDX_DFN)                          ; 1621
                              04  13 00353          BEQL    65$
              40  A9            01  90 00355 64$:     MOVB    #1, 64(IRAB)                         ; 1623
                              7E  D4 00359 65$:     CLRL    -(SP)                                ; 1627
                          0000G 30  00358          BSBW    RM$RLSBKT
                5E              04  C0 0035E          ADDL2   #4, SP
                        20  A9  D4 00361 66$:     CLRL    32(IRAB)                             ; 1628
                            53  DD 00364 67$:     PUSHL   SIZE                                 ; 1633
                        08  AE  DD 00366          PUSHL   VBN
                          0000G 30  00369          BSBW    RM$GETBKT                            ;
                5E              08  C0 0036C          ADDL2   #8, SP                               ;
                6E              50  D0 0036F          MOVL    R0, ST                               ;
                2E              6E  E8 00372          BLBS    ST, 71$                              ; 1638
    17        04  A9            15  E5 00375          BBCC    #21, 4(IRAB), 68$                    ; 1648
                54        0084  C9  D0 0037A          MOVL    132(IRAB), BDB                       ; 1651
            008C  C9        1C  A4  D0 0037F          MOVL    28(BDB), 140(IRAB)                   ; 1652
                        0084  C9  D4 00385          CLRL    132(IRAB)                            ; 1653
                              7E  D4 00389          CLRL    -(SP)                                ; 1654
                          0000G 30  0038B          BSBW    RM$RLSBKT
                5E              04  C0 0038E          ADDL2   #4, SP
            82AA  8F            6E  B1 00391 68$:     CMPW    ST, #33450                           ; 1657
                              05  13 00396          BEQL    70$
                50            6E  D0 00398 69$:     MOVL    ST, R0                               ; 1658
                              18  11 0039B          BRB     73$
              40  A9            01  90 0039D 70$:     MOVB    #1, 64(IRAB)                         ; 1660
                              C1  11 003A1          BRB     67$                                  ; 1631
              20  A9            54  D0 003A3 71$:     MOVL    BDB, 32(IRAB)                        ; 1663
                        04  AE  D4 003A7          CLRL    VBN                                  ; 1664
                56        0E  A5  9E 003AA 72$:     MOVAB   14(R5), REC_ADDR                     ; 1667
                        0094  C9  D4 003AE          CLRL    148(IRAB)                            ; 1668
                          FC53  31 003B2          BRW     1$                                   ; 0986
                5E              0C  C0 003B5 73$:     ADDL2   #12, SP                              ; 1671
                                0C  BA 003B8          POPR    #^M<R2,R3>                           ;
                                    05 003BA          RSB                                          ;
```

; Routine Size: 955 bytes,   Routine Base: RM$RMS3 + 01E9

; 1612   1672 1

```
;  1614          1673  1 GLOBAL ROUTINE RM$CSEARCH_TREE : RL$RABREG_67 =
;  1615          1674  1
;  1616          1675  1 !++
;  1617          1676  1 !
;  1618          1677  1 ! FUNCTIONAL DESCRIPTION:
;  1619          1678  1 !
;  1620          1679  1 !       This module controls the search through a tree.  If a start position
;  1621          1680  1 !       is given, the search continues from that point.  If no start position is
;  1622          1681  1 !       given, then the search begins at the root.  In the position for insert case,
;  1623          1682  1 !       when a bucket is found on a horizontal search, a record lock error is
;  1624          1683  1 !       returned from the search tree routine.  This controlling module then
;  1625          1684  1 !       restarts the search from the root to prevent deadlock.
;  1626          1685  1 !
;  1627          1686  1 ! CALLING SEQUENCE:
;  1628          1687  1 !       RM$CSEARCH_TREE()
;  1629          1688  1 !
;  1630          1689  1 ! INPUT PARAMETERS:
;  1631          1690  1 !       NONE
;  1632          1691  1 !
;  1633          1692  1 ! IMPLICIT INPUTS:
;  1634          1693  1 !       IRAB                        - address of internal RAB
;  1635          1694  1 !       IRAB[KEYBUFFER 2]           - address of search key
;  1636          1695  1 !       IRAB[IRB$B_KYSZ]            - size of key to compare(not equal to total key size if
;  1637          1696  1 !                                     generic search
;  1638          1697  1 !       IRAB[IRB$B_STOPLEVEL]       - level to stop search at
;  1639          1698  1 !       IRAB[IRB$W_SRCHFLAGS]       -
;  1640          1699  1 !       IRAB[IRB$V_POSINSERT]       - if set, this is a position for insert
;  1641          1700  1 !       IRAB[IRB$V_SRCHGT]          - if set, this is a GT approximate search
;  1642          1701  1 !       IRAB[IRB$V_SRCHGE]          - if set, this is a GE approximate search
;  1643          1702  1 !       IRAB[IRB$V_FIRST_TIM]       - if set, this is the 1ST sequential positioning
;  1644          1703  1 !                                     - after a $connect or $rewind
;  1645          1704  1 !       IRAB[IRB$V_NORLS_RNF]       - if set, do not release on RNF error
;  1646          1705  1 !       IRAB[IRB$L_CURBDB]          - if zero, then start at root
;  1647          1706  1 !       IRAB[IRB$L_CURBDB]          - if non-zero, then
;  1648          1707  1 !       REC_ADDR                    - address of record in bucket to start search at
;  1649          1708  1 !       IDX_DFN                     - address of current index descriptor
;  1650          1709  1 !
;  1651          1710  1 !
;  1652          1711  1 ! OUTPUT PARAMETERS:
;  1653          1712  1 !       NONE
;  1654          1713  1 !
;  1655          1714  1 ! IMPLICIT OUTPUTS:
;  1656          1715  1 !       REC_ADDR                    - address of index/data record which terminated search
;  1657          1716  1 !       IRAB[IRB$L_CURBDB]          - address of current BDB
;  1658          1717  1 !       IRAB[IRB$L_LOCK_BDB]        - address of level above current if locked
;  1659          1718  1 !       IRAB[IRB$V_ABOVELCKD]       - set if level above data level locked
;  1660          1719  1 !       IRAB[IRB$V_NORLS_RNF]       - always cleared
;  1661          1720  1 !
;  1662          1721  1 ! ROUTINE VALUE:
;  1663          1722  1 !       RMS$_RNF                    - record not found
;  1664          1723  1 !       RMS$_SUC                    - record found, in approximate search key
;  1665          1724  1 !                                     may not equal record/index key
;  1666          1725  1 !       Miscellaneous I/O errors
;  1667          1726  1 !
;  1668          1727  1 ! SIDE EFFECTS:
;  1669          1728  1 !       if successful, bucket is accessed
;  1670          1729  1 !       if RNF and NORLS_RNF is set, then bucket is accessed
```

```
;  1671          1730  1 !       bucket released on all other errors
;  1672          1731  1 !       permanence is set in the root's bdb, if not pos-for-insert
;  1673          1732  1 !               or there are extra buffers hanging around
;  1674          1733  1 !
;  1675          1734  1 !--
;  1676          1735  1
;  1677          1736  2     BEGIN
;  1678          1737  2
;  1679          1738  2     EXTERNAL REGISTER
;  1680          1739  2         COMMON_RAB_STR,
;  1681          1740  2         R_REC_ADDR_STR,
;  1682          1741  2         R_IDX_DFN_STR;
;  1683          1742  2
;  1684          1743  2     GLOBAL REGISTER
;  1685          1744  2         R_BDB_STR,
;  1686          1745  2         R_BKT_ADDR_STR;
;  1687          1746  2
;  1688          1747  2     IRAB[IRB$V_ABOVELCKD] = 0;          ! make sure this is clear at the start
;  1689          1748  2
;  1690          1749  2     WHILE 1
;  1691          1750  2     DO
;  1692          1751  3         BEGIN
;  1693          1752  3         BDB = .IRAB[IRB$L_CURBDB];
;  1694          1753  3
;  1695          1754  3         IF .BDB EQLU 0
;  1696          1755  3         THEN
;  1697          1756  3
;  1698          1757  3             ! If no position given, start at root
;  1699          1758  3             !
;  1700          1759  4             BEGIN
;  1701          1760  4
;  1702          1761  4             ! If the index has not been initialized, read in the index des
;  1703          1762  4             ! criptor again if it still has not been initialized, return error
;  1704          1763  4             ! Once the index descriptors are shared, this code can come out
;  1705          1764  4             ! since when the index is made, the in_core index descriptors will
;  1706          1765  4             ! be updated.
;  1707          1766  4             !
;  1708          1767  4
;  1709          1768  4             IF .IDX_DFN[IDX$V_INITIDX]
;  1710          1769  4             THEN
;  1711          1770  5                 BEGIN
;  1712          1771  5                 IRAB[IRB$V_NEW_IDX] = 1;
;  1713          1772  5
;  1714        P 1773  5                 RETURN_ON_ERROR (RMSKEY_DESC(.IDX_DFN[IDX$B_KEYREF]),
;  1715        P 1774  5                     BEGIN
;  1716        P 1775  5                     (IRAB[IRB$V_NORLS_RNF] = 0)
;  1717          1776  5                     END);
;  1718          1777  5
;  1719          1778  5                 IF .IDX_DFN[IDX$V_INITIDX]
;  1720          1779  5                 THEN
;  1721          1780  6                     (IRAB[IRB$V_NORLS_RNF] = 0;
;  1722          1781  6
;  1723          1782  6                     IF .IRAB[IRB$V_POSINSERT]
;  1724          1783  6                     THEN
;  1725          1784  7                         RETURN RMSERR(IDX)
;  1726          1785  6                     ELSE
;  1727          1786  5                         RETURN RMSERR(RNF));
```

```
;  1728          1787  5
;  1729          1788  4              END;
;  1730          1789  4
;  1731          1790  4          ! Read root and make sure it is still the root.  If not reread
;  1732          1791  4          ! prologue to obtain root VBN.
;  1733          1792  4          !
;  1734          1793  4
;  1735          1794  4          WHILE 1
;  1736          1795  4          DO
;  1737          1796  5              BEGIN
;  1738          1797  5
;  1739          1798  5              ! if pos for insert then -- if the root should turn out to be
;  1740          1799  5              ! the stoplevel then lock it
;  1741          1800  5              !
;  1742          1801  5
;  1743          1802  5              IF .IRAB[IRB$V_POSINSERT]
;  1744          1803  5              THEN
;  1745          1804  6                  BEGIN
;  1746          1805  6
;  1747          1806  6                  IF .IRAB[IRB$B_STOPLEVEL] EQL 0
;  1748          1807  6                  THEN
;  1749          1808  7                      BEGIN
;  1750          1809  7
;  1751          1810  7                      IRAB [IRB$L_VBN_LEFT] = 0;
;  1752          1811  7                      IRAB [IRB$L_VBN_RIGHT] = 0;
;  1753          1812  7                      IRAB [IRB$L_NEXT_DOWN] = 0;
;  1754          1813  7
;  1755          1814  7                      ! If the root is at level 1, we want to save
;  1756          1815  7                      ! it's VBN so that the down pointer from it can
;  1757          1816  7                      ! be checked when level 0 is reached.
;  1758          1817  7                      !
;  1759          1818  7
;  1760          1819  7                      IF .IDX_DFN [IDX$B_ROOTLEV] EQL 1
;  1761          1820  7                      THEN
;  1762          1821  7                          IRAB [IRB$L_VBN_RIGHT] = .IDX_DFN [IDX$L_ROOTVBN];
;  1763          1822  7                      END
;  1764          1823  6                  ELSE
;  1765          1824  6
;  1766          1825  6                      IF .IDX_DFN[IDX$B_ROOTLEV] EQL .IRAB[IRB$B_STOPLEVEL]
;  1767          1826  6                      THEN
;  1768          1827  6                          IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  1769          1828  6
;  1770          1829  5                  END;
;  1771          1830  5
;  1772          1831  5              ! try to get the root bucket
;  1773          1832  5              !
;  1774          1833  5
;  1775        P 1834  5              RETURN_ON_ERROR (RM$GETBKT(.IDX_DFN[IDX$L_ROOTVBN],
;  1776        P 1835  5                      .IDX_DFN[IDX$B_IDXBKTSZ]*512),
;  1777        P 1836  5                  BEGIN
;  1778        P 1837  5                  (IRAB[IRB$V_NORLS_RNF] = 0)
;  1779          1838  5                  END);
;  1780          1839  5
;  1781          1840  5              ! if it really is the root bucket, leave this loop
;  1782          1841  5              !
;  1783          1842  5
;  1784          1843  5              IF .BBLOCK[.BDB[BDB$L_ADDR], BKT$V_ROOTBKT]
```

```
;  1785          1844  5                THEN
;  1786          1845  5                    EXITLOOP;
;  1787          1846  5
;  1788          1847  5                RM$RLSBKT(0);
;  1789          1848  5                IRAB[IRB$V_NEW_IDX] = 1;
;  1790          1849  5
;  1791        P 1850  5                RETURN_ON_ERROR (RM$KEY_DESC(.IDX_DFN[IDX$B_KEYREF]),
;  1792        P 1851  5                    BEGIN
;  1793        P 1852  5                    (IRAB[IRB$V_NORLS_RNF] = 0)
;  1794          1853  5                    END);
;  1795          1854  5
;  1796          1855  4                END;
;  1797          1856  4
;  1798          1857  4            IRAB[IRB$L_CURBDB] = .BDB;
;  1799          1858  4
;  1800          1859  4            IF NOT .IRAB[IRB$V_POSINSERT]
;  1801          1860  4                OR
;  1802          1861  5                (.IDX_DFN[IDX$B_KEYREF] LSSU .IFAB[IFB$B_EXTRABUF])
;  1803          1862  4            THEN
;  1804          1863  4                BDB[BDB$V_PRM] = 1;
;  1805          1864  4
;  1806          1865  4            REC_ADDR = .BDB[BDB$L_ADDR] + BKT$C_OVERHDSZ;
;  1807          1866  4            IRAB[IRB$L_REC_COUNT] = 0;
;  1808          1867  4            END
;  1809          1868  3        ELSE
;  1810          1869  4            BEGIN
;  1811          1870  4
;  1812          1871  4            IF .IRAB[IRB$V_FIRST_TIM]
;  1813          1872  4            THEN
;  1814          1873  4                RETURN RMSERR (BUG);
;  1815          1874  4
;  1816          1875  4            BKT_ADDR = .BDB[BDB$L_ADDR]
;  1817          1876  3            END;
;  1818          1877  3
;  1819          1878  4        BEGIN
;  1820          1879  4
;  1821          1880  4        LOCAL
;  1822          1881  4            STS;
;  1823          1882  4
;  1824          1883  4        STS = RM$SEARCH_TREE();
;  1825          1884  4
;  1826          1885  5        IF .STS<0, 16> NEQU RMSERR(RLK)
;  1827          1886  4        THEN
;  1828          1887  5            BEGIN
;  1829          1888  5
;  1830          1889  6            IF .STS<0, 16> EQL RMSERR(RNF)
;  1831          1890  5                AND
;  1832          1891  5                NOT .IRAB[IRB$V_NORLS_RNF]
;  1833          1892  5            THEN
;  1834          1893  6                BEGIN
;  1835          1894  6                IRAB[IRB$L_CURBDB] = 0;
;  1836          1895  6                RM$RLSBKT(0)
;  1837          1896  5                END;
;  1838          1897  5
;  1839          1898  5            IRAB[IRB$V_NORLS_RNF] = 0;
;  1840          1899  5
;  1841          1900  5            RETURN .STS
```

```
;  1842          1901  5            END
;  1843          1902  3        END;
;  1844          1903  3
;  1845          1904  3        IRAB[IRB$B_SPL_BITS] = 0;       ! clear dups_seen, empt_seen when
;  1846          1905  3                                        ! going around again from data level
;  1847          1906  3        END
;  1848          1907  3
;  1849          1908  1    END;                                 ! end of routine RM$CSEARCH_TREE


                                      30  BB 00000 RM$CSEARCH_TREE::
                                                    PUSHR   #^M<R4,R5>                      ; 1673
                    06  A9                20  8A 00002           BICB2   #32, 6(IRAB)            ; 1747
                        54        20      A9  D0 00006 1$:       MOVL    32(IRAB), BDB           ; 1752
                                          03  13 0000A           BEQL    2$                      ; 1754
                                      00B3  31 0000C           BRW     12$
                2C      1C  A7            04  E1 0000F 2$:       BBC     #4, 28(IDX_DFN), 4$     ; 1768
                    42  A9                08  88 00014           BISB2   #8, 66(IRAB)            ; 1771
                        7E        21      A7  9A 00018           MOVZBL  33(IDX_DFN), -(SP)      ; 1776
                                    0000G 30  0001C           BSBW    RM$KEY_DESC
                        5E                04  C0 0001F           ADDL2   #4, SP
                        78                50  E9 00022           BLBC    STATUS, 7$
                16      1C  A7            04  E1 00025           BBC     #4, 28(IDX_DFN), 4$     ; 1778
                    42  A9                20  8A 0002A           BICB2   #32, 66(IRAB)           ; 1780
                        07        42      A9  E9 0002E           BLBC    66(IRAB), 3$            ; 1782
                        50      855C      8F  3C 00032           MOVZWL  #34140, R0              ; 1786
                                          68  11 00037           BRB     8$
                        50      82B2      8F  3C 00039 3$:       MOVZWL  #33458, R0
                                          61  11 0003E           BRB     8$
                        26        42      A9  E9 00040 4$:       BLBC    66(IRAB), 6$            ; 1802
                                  41      A9  95 00044           TSTB    65(IRAB)                ; 1806
                                          16  12 00047           BNEQ    5$
                                    0088  C9  7C 00049           CLRQ    136(IRAB)               ; 1810
                                    0090  C9  D4 0004D           CLRL    144(IRAB)               ; 1812
                        01              15  A7  91 00051           CMPB    21(IDX_DFN), #1         ; 1819
                                          13  12 00055           BNEQ    6$
                    008C    C9        18  A7  D0 00057           MOVL    24(IDX_DFN), 140(IRAB)  ; 1821
                                          0B  11 0005D           BRB     6$                      ; 1806
                    41      A9        15  A7  91 0005F 5$:       CMPB    21(IDX_DFN), 65(IRAB)   ; 1825
                                          04  12 00064           BNEQ    6$
                    40      A9                01  90 00066           MOVB    #1, 64(IRAB)            ; 1827
                            50        16  A7  9A 0006A 6$:       MOVZBL  22(IDX_DFN), R0         ; 1838
                7E          50              09  78 0006E           ASHL    #9, R0, -(SP)
                                    18  A7  DD 00072           PUSHL   24(IDX_DFN)
                                  0000G 30  00075           BSBW    RM$GETBKT
                        5E                08  C0 00078           ADDL2   #8, SP
                        1F                50  E9 0007B           BLBC    STATUS, 7$
                        50        18      A4  D0 0007E           MOVL    24(BDB), R0             ; 1843
                1C      0D  A0            01  E0 00082           BBS     #1, 13(R0), 9$
                                          7E  D4 00087           CLRL    -(SP)                   ; 1847
                                  0000G 30  00089           BSBW    RM$RLSBKT
                    42  A9                08  88 0008C           BISB2   #8, 66(IRAB)            ; 1848
                        6E        21      A7  9A 00090           MOVZBL  33(IDX_DFN), (SP)       ; 1853
                                  0000G 30  00094           BSBW    RM$KEY_DESC
```

```
                5E              04  C0 00097          ADDL2   #4, SP                                          ;
                A3              50  E8 0009A          BLBS    STATUS, 4$                                      ;
             42 A9              20  8A 0009D 7$:      BICB2   #32, 66(IRAB)                                  ;
                                62  11 000A1 8$:      BRB     17$                                             ;
             20 A9              54  D0 000A3 9$:      MOVL    BDB, 32(IRAB)                                   ; 1857
                08          42  A9  E9 000A7          BLBC    66(IRAB), 10$                                   ; 1859
          00B6  CA          21  A7  91 000AB          CMPB    33(IDX_DFN), 182(IFAB)                          ; 1861
                                04  1E 000B1          BGEQU   11$                                             ;
             0A A4              08  88 000B3 10$:     BISB2   #8, 10(BDB)                                    ; 1863
          56 18 A4              0E  C1 000B7 11$:     ADDL3   #14, 24(BDB), REC_ADDR                          ; 1865
                          0094  C9  D4 000BC          CLRL    148(IRAB)                                       ; 1866
                                10  11 000C0          BRB     14$                                             ; 1754
          07 42 A9              06  E1 000C2 12$:     BBC     #6, 66(IRAB), 13$                               ; 1871
                50        8434  8F  3C 000C7          MOVZWL  #33844, R0                                      ; 1873
                                37  11 000CC          BRB     17$                                             ;
                55          18  A4  D0 000CE 13$:     MOVL    24(BDB), BKT_ADDR                               ; 1875
                            FB70  30  00D2 14$:       BSBW    RM$SEARCH_TREE                                  ; 1883
                51              50  D0 000D5          MOVL    R0, STS                                         ;
             82AA 8F            51  B1 000D8          CMPW    STS, #33450                                     ; 1885
                                20  13 000DD          BEQL    16$                                             ;
             82B2 8F            51  B1 000DF          CMPW    STS, #33458                                     ; 1889
                                10  12 000E4          BNEQ    15$                                             ;
          0B 42 A9              05  E0 000E6          BBS     #5, 66(IRAB), 15$                               ; 1891
                          20    A9  D4 000EB          CLRL    32(IRAB)                                        ; 1894
                                7E  D4 000EE          CLRL    -(SP)                                           ; 1895
                           0000G    30 00F0          BSBW    RM$RLSBKT                                       ;
                5E              04  C0 00F3          ADDL2   #4, SP                                          ;
             42 A9              20  8A 00F6 15$:      BICB2   #32, 66(IRAB)                                  ; 1898
                50              51  D0 00FA          MOVL    STS, R0                                         ; 1900
                                06  11 00FD          BRB     17$                                             ;
                          44    A9  94 00FF 16$:      CLRB    68(IRAB)                                        ; 1904
                             FF01   31 00102          BRW     1$                                              ; 1749
                                30  BA 00105 17$:      POPR    #^M<R4,R5>                                      ; 1908
                                    05 00107          RSB                                                     ;

; Routine Size:  264 bytes,    Routine Base:  RM$RMS3 + 05A4

; 1850          1909  1
```

```
;  1852          1910  1 GLOBAL ROUTINE RM$V3_VBN : RL$RABREG_567 =
;  1853          1911  1 !+++
;  1854          1912  1 !
;  1855          1913  1 ! RM$V3_VBN
;  1856          1914  1 !
;  1857          1915  1 !       This routine returns the VBN associated with record number
;  1858          1916  1 !       'rec_count'. It is only useful on Version 3.0 index buckets.
;  1859          1917  1 !
;  1860          1918  1 !---
;  1861          1919  2 BEGIN
;  1862          1920  2
;  1863          1921  2 EXTERNAL REGISTER
;  1864          1922  2     R_IRAB_STR,
;  1865          1923  2     R_IFAB_STR,
;  1866          1924  2     R_IDX_DFN_STR,
;  1867          1925  2     R_BKT_ADDR_STR,
;  1868          1926  2     R_REC_ADDR_STR;
;  1869          1927  2
;  1870          1928  2 LOCAL
;  1871          1929  2     VBN,                    ! temporary storage
;  1872          1930  2     FIRST_VBN,              ! addr of first VBN in bucket
;  1873          1931  2     VBN_SIZE;               ! size of VBNs in this index bucket
;  1874          1932  2
;  1875          1933  2 VBN_SIZE = (.BKT_ADDR[BKT$V_PTR_SZ]) + 2;
;  1876          1934  2 FIRST_VBN = .BKT_ADDR + (.IDX_DFN[IDX$B_IDXBKTSZ] * 512) - 4 - .VBN_SIZE;
;  1877          1935  2 VBN = .(.FIRST_VBN - (.VBN_SIZE * .IRAB[IRB$L_REC_COUNT]));
;  1878          1936  2 RETURN .VBN <0,.VBN_SIZE ^*3>
;  1879          1937  1 END;


                                                52  DD 00000 RM$V3_VBN::
                                                        PUSHL   R2                                      ;  1910
        51      0D   A5             02          03  EF 00002           EXTZV   #3, #2, 13(BKT_ADDR), VBN_SIZE  ;  1933
                                    51          02  C0 00008           ADDL2   #2, VBN_SIZE
                                    50      16  A7  9A 0000B           MOVZBL  22(IDX_DFN), R0                 ;  1934
                50                  50          09  78 0000F           ASHL    #9, R0, R0
                                    50          55  C0 00013           ADDL2   BKT_ADDR, R0
                                    50          51  C2 00016           SUBL2   VBN_SIZE, R0
                                    50          04  C2 00019           SUBL2   #4, FIRST_VBN
                52                  51  0094    C9  C5 0001C           MULL3   148(IRAB), VBN_SIZE, R2         ;  1935
                                    50          52  C2 00022           SUBL2   R2, R0
                                    52          60  D0 00025           MOVL    (R0), VBN
                                    51          08  C4 00028           MULL2   #8, R1                          ;  1936
        50              52          51          00  EF 0002B           EXTZV   #0, R1, VBN, R0
                                                04  BA 00030           POPR    #^M<R2>                         ;  1937
                                                    05 00032           RSB

; Routine Size:  51 bytes,    Routine Base:  RM$RMS3 + 06AC

;  1880          1938  1 END
;  1881          1939  0 ELUDOM
```

```
;                                PSECT SUMMARY

;       Name                    Bytes                           Attributes

;  RM$RMS3                       1759  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       115         3       154          00:00.4


;                                COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3SRCHKY/OBJ=OBJ$:RM3SRCHKY MSRC$:RM3SRCHKY/UPDATE=(ENH$:RM3SRCHKY)

; Size:         1759 code + 0 data bytes
; Run Time:       00:44.8
; Elapsed Time:   01:23.0
; Lines/CPU Min:   2597
; Lexemes/CPU-Min: 16722
; Memory Used:  401 pages
; Compilation Complete
```

RM3SSIDR
LIS

RM3UPSIDX
LIS

RM3SRCHKY
LIS

RM3UPDDEL
LIS

RM3UPDATE
LIS