```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     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  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

FILEID**RM3SRCHKY


```
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
```

```
0 MODULE RM3SRCHKY (LANGUAGE (BLISS32) ,
0                   IDENT = 'V04-000'
0                   ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 !
1 ! FACILITY:      RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
1 !
1 ! ABSTRACT:
1 !                This module starts at current bucket and searchs to stop level
1 !
1 !
1 ! ENVIRONMENT:
1 !
1 !               VAX/VMS OPERATING SYSTEM
1 !
1 !--
1
1 !
1 ! AUTHOR:        D. H. Gillespie     CREATION DATE:          18-APR-78  9:39
1 !
1 ! MODIFIED BY:
1 !
1 !       V03-011 TMK0004         Todd M. Katz            03-Apr-1983
1 !               Add support for RU record space reclamation. This involves
1 !               modifications to RM$SEARCH_TREE as follows. Whenever RMS
1 !               has write access to a file that is marked RU Journallable,
1 !               and is currently positioning to a primary data bucket but
1 !               not for modification, then RMS wants to set the CSH$V_LOCK
1 !               flag so that the bucket will be EXclusively locked in case RU
1 !               record reclamation takes place on the bucket. In addition, RMS
1 !               wants to EXclusively lock all SIDR buckets whenever it has
1 !               write access to the file.
```

```
!
!       V03-010 MCN0005         Maria del C. Nasr       15-Mar-1983
!               More linkages reorganization
!
!       V03-009 MCN0004         Maria del C. Nasr       28-Feb-1983
!               Reorganize linkages
!
!       V03-008 TMK0003         Todd M. Katz            03-Feb-1983
!               Add support for Recovery Unit Journalling and RU ROLLBACK
!               Recovery of ISAM files. Make a change to SEARCH_V2. Do not set
!               the bit IRB$V_DUPS_SEEN if the current primary data record is
!               either marked RU_DELETE or DELETED. Previously just a check for
!               DELETED was being made.
!
!       V03-007 TMK0002         Todd M. Katz            09-Sep-1982
!               The field IRB$B_SRCHFLAGS has been changed to a word in size.
!               Fix all references to it.
!
!               Add support for prologue 3 SIDRs. This includes the removal of
!               misleading comment statements.
!
!               In the local routine SEARCH_V2, when a search key is found to
!               be equal to a key of a record, set the IRAB bit IRB$V_DUP_KEY.
!
!
!       V03-006 KBT0297         Keith B. Thompson       24-Aug-1982
!               Reorganize psects
!
!       V03-005 MCN0003         Maria del C. Nasr       29-Jun-1982
!               Allow keys of different data types other than string
!               in prologue 3 files.  Change all CH$COMPARE calls to
!               RM$COMPARE_KEY to compare keys taking into consideration
!               the different data types.
!
!       V03-004 TMK0001         Todd M. Katz            25-May-1982
!               Eliminate unneccesary linkage declaration for RM$RECORD_KEY.
!
!       V03-003 MCN0002         Maria del C. Nasr       25-Mar-1982
!               Use macro to calculate key buffer address.
!
!       V03-002 CDS0008         C Saether               23-Mar-1982
!               Restore BKT_ADDR when BDB is restored when leaving
!               block that confirms down pointer from level 1.
!               This was causing the check for level 0 to fail and
!               not lock the next bucket when positioning horizontally
!               at level 0.
!
!       V03-001 CDS0007         C Saether               3-Mar-1982
!               Fix bug where horizontal reads at the stoplevel were
!               not locking the bucket on positions for insert or
!               delete.  This was working correctly for position for
!               insert at level 0 only (which is the normal place
!               where horizontal reads would take place).
!               The changes in RM$CACHE and RM$RELEASE have made them
!               sensitive to this error.
!
!       V02-039 CDS0006         C Saether               20-Jan-1982
```

```
1 !                 Set VBN_LEFT for scan forward when not locking.
1 !                 Clear IRB$L_REC_COUNT before calling RM$SRCH_BY_KEY
1 !                 in the horizontal scan logic.
1 !                 Once data level horizontal scan has started, do not
1 !                 drop back into outer loop anymore.
1 !
1 !         V02-038 CDS0005         C Saether               14-Jan-1982
1 !                 Put back logic to check index down pointer after
1 !                 positioning to level on insert.  It's really necessary.
1 !
1 !         V02-037 CDS0004         C Saether               31-Dec-1981
1 !                 Fix error path botched in V02-036.
1 !                 Put back retry on rlk error going down tree inadvertently
1 !                 removed by V02-036.
1 !
1 !         V02-036 CDS0003         C Saether               29-Dec-1981
1 !                 Remove logic to check down pointer before going
1 !                 across on position for insert.
1 !                 Modify horizontal scan at data level on position for
1 !                 insert to release bucket prior to going ahead.  This
1 !                 eliminates need for special algorithms to avoid deadlock
1 !                 with rrv updating by other streams.  Previous algorithm
1 !                 could get into infinite loop with another process.
1 !
1 !         V02-035 PSK0009         P Knibbe                07-Oct-1981
1 !                 Make sure all calls to RM$RECORD_VBN check for
1 !                 prologue version.
1 !
1 !         V02-034 CDS0002         C Saether               29-Sep-1981
1 !                 Make check for exclusive lock before setting abovelckd.
1 !
1 !         V02-033 CDS0001         C Saether               20-Aug-1981
1 !                 Remove references to BDB$L_OWN.
1 !
1 !         V02-032 PSK0008         Paulina S. Knibbe       08-Aug-1981
1 !                 Remove all references to SPLCTX
1 !
1 !         V02-031 PSK0007         Paulina S. Knibbe       30-Jul-1981
1 !                 Remove support for truncated index keys.
1 !
1 !         V02-030 PSK0006         Paulina S. Knibbe       12-Jun-1981
1 !                 1) Make RM$V3_VBN a global routine
1 !                 2) Save LST_NCMP when bouncing back and forth between
1 !                    two buckets.
1 !                 3) Add logic to support horizontal processing in V3
1 !                    compresses indexes.
1 !                 4) Add RM$CHK_HORIZ
1 !                 5) Change calls to COMP_ASCII to be CH$COMPARE
1 !
1 !         V02-029 PSK0005         Paulina S. Knibbe       28-May-1981
1 !                 Fix problem on GT search. Make lock-above optimization
1 !                 work for prologue three files.
1 !
1 !         V02-028 PSK0004         Paulina S. Knibbe       03-May-1981
1 !                 Add support for compressed keys in index, SIDR and data
1 !                 buckets.
1 !
```

```
!          V02-027 PSK0003          Paulina S. Knibbe        30-Apr-1981
!                  Fix problem when key goes past range in bucket
!
!          V02-026 MCN0001          Maria del C. Nasr        23-Apr-1981
!                  Add support for non-compressed keys at the primary data
!                  level of prologue 3 files.
!
!          V02-025 PSK0002          Paulina S. Knibbe        30-Mar-1981
!                  Add support for random searching of PLG3 buckets with
!                  fixed length keys.
!
!          V02-024 PSK0001          Paulina S. Knibbe        19-Mar-1981
!                  Change index bucket VBN processing to handle VBN lists
!                  at end of bucket
!
!          V02-023 REFORMAT         Keith B. Thompson        23-Jul-1980
!
! REVISION HISTORY:
!
!          Christian Saether,       27-JUL-78  10:48
!          X0002 - SRCH_BY_KEY changed to store last record address for
!                  ADD_TO_ARRAY
!
!          Christian Saether,        9-AUG-78  11:55
!          X0003 - never return RNF on POSINSERT
!
!          Wendy Koenig,    11-AUG-78  14:10
!          X0004 - only lock the root if it is the lock-above or stoplevel level
!
!          Christian Saether,       14-AUG-78  10:21
!          X0005 - add code to disable RNF on position for insert
!
!          Christian Saether,       24-AUG-78  9:39
!          X0006 - Add POSDELETE to SEARCH_TREE
!
!          Christian Saether,       24-AUG-78  19:41
!          X0007 - Correct logic error horizontal positioning on insert across
!                  empty buckets
!
!          Christian Saether,       26-SEP-78  15:57
!          X0008 - never do lock_above on position for delete
!
!          D. H. Gillespie,          6-OCT-78  17:34
!          X0009 - fix bug when positioning for insert and entire file is empty
!                  from where search started and end
!
!          Christian Saether,        9-OCT-78  12:10
!          X0010 - reverse usage of EMPT_SEEN and EMPTY_BKT so EMPTY_BKT is not
!                  clobbered while searching index levels
!
!          Wendy Koenig,    24-OCT-78  14:03
!          X0011 - make changes caused by sharing conventions
!
!          Christian Saether,       27-OCT-78  15:09
!          X0012 - do not set DUPS_SEEN if the record is deleted on position for
!                  insert
!
```

```
!       Christian Saether,      12-DEC-78  20:17
!       X0013 - set ADUP_SEEN always when passing dupes - this has been taken
!               out 12/18/78
!
!       Christian Saether,      15-DEC-78  11:17
!       X0014 - mask split_bits in csearch_tree before calling search_tree
!
!       Christian Saether,      9-JAN-79  12:08
!       X0015 - on lock error going horizontal, reread bucket with lock to
!               force stall
!
!       Christian Saether,      15-JAN-79  18:22
!       X0016 - modify lockabove logic to avoid deadlocks
!
!       Christian Saether,      20-JAN-79  20:18
!       X0017 - confirm level 1 down pointer before going across level 0
!
!       Wendy Koenig,   23-JAN-79  14:21
!       X0018 - zero MIDX_TMP1
!
!       Christian Saether,      24-JAN-79  11:25
!       X0019 - set up bkt_addr after releasing empty bucket when going across
!               at data level
!
!       Christian Saether,      26-JAN-79  9:49
!       X0020 - only clear splits bits when coming around from data level,
!               take out going across at data level kluge
!
!       Christian Saether,      6-FEB-79  17:42
!       X0021 - do not look at owner of bdb to make lockabove decision
!
!       Christian Saether,      14-FEB-79  14:05
!       X0022 - always go across data level primary key dupes
!
!*****

LIBRARY 'RMSLIB:RMS';

REQUIRE 'RMSSRC:RMSIDXDEF';

! define default psects for code
!
PSECT
    CODE = RM$RMS3(PSECT_ATTR),
    PLIT = RM$RMS3(PSECT_ATTR);

!

MACRO
    L_HIGH_KEY =
        RL$HIGH_KEY = JSB()%;


! Linkages
!

LINKAGE
```

```
1         L_CACHE,
1         L_COMPARE_KEY,
1         L_HIGH_KEY,
1         L_PRESERVE1,
1         L_RABREG_4567,
1         L_RABREG_457,
1         L_RABREG_567,
1         L_RABREG_67,
1         L_RABREG_7,
1         L_REC_OVHD;
1
1  !
1  ! Forward Routines
1  !
1
1  FORWARD ROUTINE
1         RM$V3_VBN             : RL$RABREG_567;
1
1  !
1  ! External Routines
1  !
1
1  EXTERNAL ROUTINE
1         RM$CACHE              : RL$CACHE ADDRESSING_MODE( GENERAL ),
1         RM$CNTRL_ADDR         : RL$RABREG_567,
1         RM$COMPARE_KEY        : RL$COMPARE_KEY,
1         RM$GETBKT             : RL$RABREG_457,
1         RM$KEY_DESC           : RL$RABREG_7,
1         RM$REC_OVHD           : RL$REC_OVHD,
1         RM$RECORD_VBN         : RL$PRESERVE1,
1         RM$RLSBKT             : RL$PRESERVE1,
1         RM$SRCH_CMPR          : RL$RABREG_567;
1
1  LITERAL
1         LS = -1,
1         GT = 1,
1         EQ = 0;
```

```
1 ROUTINE SEARCH_FIX (GOAL) : RL$RABREG_567 =
1
1 !++
1 !
1 ! SEARCH_FIX
1 !
1 !       This routine searches a PLG3 fixed length key index bucket from
1 !       the current record address to the end of the bucket for an index
1 !       record equal or greater than the input search key.
1 !
1 ! CALLING SEQUENCE:
1 !       SEARCH_FIX
1 !
1 ! INPUT PARAMETERS:
1 !
1 !       GOAL - 0 - be satisfied with an equal match
1 !            - 1 - position past an equal match (search for GT match)
1 !
1 ! IMPLICIT INPUTS:
1 !       REC_ADDR                    - address of record in bucket to begin search on
1 !       BKT_ADDR                    - address of current bucket
1 !       IDX_DFN                     - address of index descriptor for current key of reference
1 !       IRAB                        - address of internal RAB
1 !       IRAB[IRB$V_SRCHGT]          - if set, search for index record gt search key
1 !       IRAB[IRB$V_POSINSERT]       - if set, search for position to insert record
1 !       IRAB[KEY_BUFFER_2]          - address of search key
1 !       IRAB[IRB$B_KEYSZ]           - size of key to compare
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       REC_ADDR          - if EQ, address of index record equal to search key
1 !                         - if GT, at end of record data in bucket
1 !                         - if LS, address of index record greater than search key
1 !       IRAB [IRB$L_REC_COUNT]  - Number of the record who address is in REC_ADDR
1 !       [DUPS_SEEN]       - Set if goal is gt and we found an eq match
1 !
1 ! ROUTINE VALUE:
1 !       R0                -  0, search key = index record
1 !                         - -1, search key < index record
1 !                         - +1, search key > index record
1 !
1 ! SIDE EFFECTS:
1 !
1 !--
2 BEGIN
2
2 EXTERNAL REGISTER
2     R_IRAB_STR,
2     R_IFAB_STR,
2     R_IDX_DFN_STR,
2     R_BKT_ADDR_STR,
2     R_REC_ADDR_STR;
2
2 LOCAL
2     STATUS,             ! place to hold results of compare
```

```
	RSTART,			! address of first record in range
	REND,			! address of last record in range
	EOB;			! address past last record in bucket

EOB = .BKT_ADDR + .BKT_ADDR [BKT$W_FREESPACE];
RSTART = .REC_ADDR;
REND = .EOB;

! Do a binary search of fixed length index records
!

WHILE 1 DO
	BEGIN
	LOCAL
	    SIZE;		! Number of characters in range

	BUILTIN
	    AP;

	SIZE = .REND - .RSTART;
	REC_ADDR = (.RSTART + (.SIZE^-1) / .IDX_DFN[IDX$B_KEYSZ] * .IDX_DFN[IDX$B_KEYSZ]);
	AP = 3;			! Contiguous compare
	STATUS = RM$COMPARE_KEY (.REC_ADDR, KEYBUF_ADDR(2), .IRAB [IRB$B_KEYSZ]);

	CASE .STATUS FROM LS TO GT OF
	    SET
	    [LS]:	! Search key is < record in bucket
			! prepare to search low half of bucket
			!
			BEGIN
			REND = .REC_ADDR;

			IF .REND EQL .RSTART
			THEN

			    ! Only one record left and we just compared it
			    !
			    EXITLOOP;
			END;

	    [EQ]:	! Search key = record in bucket
			!
			!	GENERIC SEARCH		GOAL IS EQ MATCH
			!	    yes			    yes			search previous
			!	    yes			    no			search next
			!	    no			    yes			return
			!	    no			    no			search next
			!
			BEGIN

			IF .GOAL EQL EQ
			THEN
			    IF .IRAB [IRB$B_KEYSZ] EQL .IDX_DFN [IDX$B_KEYSZ]
			    THEN
				EXITLOOP
			    ELSE
```

```
                                ! generic equal match. we must postion to the previous
                                ! record in order to find first equal match
                                !
                               BEGIN

                               IF (.REND - .RSTART) EQL .IDX_DFN [IDX$B_KEYSZ]
                               THEN
                                   EXITLOOP;

                               IF (.REND - .RSTART) EQL (.IDX_DFN [IDX$B_KEYSZ]^1)
                               THEN
                                   REND = .REC_ADDR
                               ELSE
                                   REND = .REC_ADDR + .IDX_DFN [IDX$B_KEYSZ];
                               END

                   ELSE

                       ! Search past the equal match
                       !
                       BEGIN

                       ! If there is another record in the bucket, compare it
                       !
                       REC_ADDR = .REC_ADDR + .IDX_DFN[IDX$B_KEYSZ];

                       IF .REC_ADDR EQL .EOB
                       THEN
                           STATUS = GT
                       ELSE
                           STATUS = LS;

                       EXITLOOP;
                       END
                   END;

           [GT]:   ! Search key is > record in bucket
                   !
                   BEGIN
                   REC_ADDR = .REC_ADDR + .IDX_DFN[IDX$B_KEYSZ];

                   IF (.REND - .RSTART) LEQ (.IDX_DFN[IDX$B_KEYSZ])^1
                   THEN

                       ! We just compared the last record in the range
                       ! If this is the last record in the bucket, return
                       ! with a GT status.
                       !

                       IF .REND EQL .EOB
                       THEN
                           EXITLOOP
                       ELSE
                           ! We are not at the end of bucket so we know that
                           ! we must have done an earlier compare against the
                           ! low record of the next range and got an LS result.
                           ! Return that record.
```

```
4                           !
5                           BEGIN
5                           STATUS = LS;
5                           EXITLOOP;
4                           END;
4
4                   ! Prepare to search high half of bucket
4                   !
4                   RSTART = .REC_ADDR;
3                   END;
3           TES;
2       END;
2
2 ! We now have a record of some sort.
2 ! Compute the record count so that we can later pick up the VBN portion
2 ! of the record from the end of the bucket
2 !
2 IRAB[IRB$L_REC_COUNT] = (.REC_ADDR-.BKT_ADDR-BKT$C_OVERHDSZ)/.IDX_DFN[IDX$B_KEYSZ];
2
2 RETURN .STATUS;
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$RECORD_VBN, RM$RLSBKT
                                                        .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 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
```

```
1 ROUTINE SEARCH_V2 (GOAL) : RL$RABREG_567 =
1 !+++
1 !
1 ! SEARCH_V2
1 !
1 !      This routine searches a prologue two bucket or a prologue 3 data level
1 !      bucket (primary or SIDR) with non-compressed keys for a record whose
1 !      value is equal or greater than the search key.
1 !
1 ! INPUT PARAMETER
1 !
1 !      GOAL      - 0, return if you find an equal or greater than match
1 !                - 1, return only with a greater than match
1 !
1 ! IMPLICIT INPUTS:
1 !      REC_ADDR                  - address of record in bucket to begin search on
1 !      BKT_ADDR                  - address of current bucket
1 !      IDX_DFN                   - address of index descriptor for current key of reference
1 !      IRAB                      - address of internal RAB
1 !      IRAB[IRB$V_SRCHGT]        - if set, search for index/data record gt search key
1 !      IRAB[IRB$V_POSINSERT]     - if set, search for position to insert record
1 !      IRAB[KEY_BUFFER_2]        - address of search key
1 !      IRAB[IRB$B_KEYSZ]         - size of key to compare
1 !      IFAB[IFB$B_PLG_VER]       - prologue version number
1 !
1 ! OUTPUT PARAMETERS:
1 !      NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !      REC_ADDR          - if EQ, address of index/record equal to search key
1 !                        - if GT, at end of record data in bucket
1 !                        - if LS, address of index/record greater than search key
1 !      [ DUPS_SEEN ]     - set if duplicates seen when SRCHGT set
1 !      [ DUP_KEY ]       - set if a duplicate key is seen when SRCHGT is set
1 !      IRB$L_LST_NCMP    - Address of last record in bucket if search key > data record
1 !
1 !
1 ! ROUTINE VALUE:
1 !      R0                - 0, search key = index/data record
1 !                        - -1, search key < index/data record
1 !                        - 1,  search key > index/data record
1 !
1 ! SIDE EFFECTS:
1 !      RRV are skipped
1 !      AP is clobbered
1 !
1 !--
2 BEGIN
2
2 EXTERNAL REGISTER
2      R_IRAB_STR,
2      R_IFAB_STR,
2      R_IDX_DFN_STR,
2      R_BKT_ADDR_STR,
2      R_REC_ADDR_STR;
2
2 BUILTIN
```

```
    AP;

LOCAL
    OVHD,
    TOTAL_SIZE,
    STATUS,
    EOB;

EOB = .BKT_ADDR + .BKT_ADDR [BKT$W_FREESPACE];

WHILE .REC_ADDR LSSA .EOB
DO
    BEGIN

    AP = 3;  ! Initialize for a contiguous compare

    BEGIN

    LOCAL
        REC_SIZE;

    ! Search for the 1st record whose key is greater than or
    ! equal to the search key
    !

    IF (REC_SIZE = .BKT_ADDR [BKT$B_LEVEL]) EQL 0
    THEN

        ! We have a data record, check if it is an RRV
        !
        BEGIN

        IF .BKT_ADDR[BKT$B_INDEXNO] EQLU 0
            AND
           .REC_ADDR [IRC$V_RRV]
        THEN
            RETURN GT;

        ! Nope. Do the setup for compare
        !

        IF .IDX_DFN [IDX$B_KEYREF] NEQ 0
        THEN

            ! We have a SIDR data record, set REC_SIZE to -1 as flag
            !
            REC_SIZE = .REC_SIZE - 1

        ELSE

            ! We have a primary data record, set AP to 2 as flag
            !
            AP = .AP - 1;
        END;

    ! Now get the number of bytes of overhead for this record
    !
```

```
    OVHD = RM$REC_OVHD(.REC_SIZE; REC_SIZE);
    TOTAL_SIZE = .OVHD + .REC_SIZE;
    END; ! of block defining rec_size

    ! Do the comparison (finally).  But first set AP to contiguous compare
    ! again, since the primary key is extracted at the front of the record.
    !

    IF .IFAB[IFB$B_PLG_VER] EQLU PLG$C_VER_3
    THEN
        AP = 3;

    STATUS = RM$COMPARE_KEY (.OVHD + .REC_ADDR,
                             KEYBUF_ADDR(2),
                             .IRAB [IRB$B_KEYSZ]);

    IF .STATUS LSS 0
    THEN
        RETURN .STATUS;

    IF .STATUS EQL 0
    THEN
        IF .GOAL EQL 0
        THEN
            RETURN EQ
        ELSE
            ! We have an equal match but were looking for a GT match.
            ! Go get the next record and continue comparison (this will
            ! position past all duplicates).
            !
            BEGIN
            IRAB[IRB$V_DUP_KEY] = 1;
            IRAB [IRB$L_LST_REC] = .REC_ADDR;

            IF  (.BKT_ADDR[BKT$B_LEVEL] EQLU 0
                         AND
                        .BKT_ADDR[BKT$B_INDEXNO] NEQU 0)
                  OR
                NOT (.REC_ADDR[IRC$V_DELETED]
                        OR
                      .REC_ADDR[IRC$V_RU_DELETE])
            THEN
                IRAB [IRB$V_DUPS_SEEN] = 1;
            END;

    REC_ADDR = .REC_ADDR + .TOTAL_SIZE;
    END;

! We have a GT match. Save the address of the last record in
! the bucket (we are positioned past it now).
!
IRAB [IRB$L_LST_NCMP] = .REC_ADDR - .TOTAL_SIZE;

RETURN GT
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
```

RM
V0

```
1 GLOBAL ROUTINE RM$SRCH_BY_KEY : RL$RABREG_567 =
1 !++
1 !
1 !     RM$SRCH_BY_KEY
1 !
1 !         This routine searches a bucket from the current record address
1 !         for an index/data record equal or greater than the input search key
1 !
1 ! CALLING SEQUENCE:
1 !         RM$SRCH_BY_KEY()
1 !
1 ! INPUT PARAMETERS:
1 !         NONE
1 !
1 ! IMPLICIT INPUTS:
1 !         REC_ADDR                  - address of record in bucket to begin search on
1 !         BKT_ADDR                  - address of current bucket
1 !         IDX_DFN                   - address of index descriptor for current key of reference
1 !         IRAB                      - address of internal RAB
1 !         IRAB[IRB$V_SRCHGT]        - if set, search for index/data record gt search key
1 !         IRAB[IRB$V_POSINSERT]     - if set, search for position to insert record
1 !         IRAB[KEY_BUFFER_2]        - address of search key
1 !         IRAB[IRB$B_KEYSZ]         - size of key to compare
1 !
1 ! OUTPUT PARAMETERS:
1 !         NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !         REC_ADDR          - if EQ, address of index/record equal to search key
1 !                           - if GT, at end of record data in bucket
1 !                           - if LS, address of index/record greater than search key
1 !         IRAB [ EMPT_SEEN ] - set if no data records encountered
1 !              [ DUPS_SEEN ] - set if duplicates seen when SRCHGT set
1 !              [ DUP_KEY ]   - set if duplicate key seen when SRCHGT is set
1 !
1 ! ROUTINE VALUE:
1 !         R0                - 0, search key = index/data record
1 !                           - -1, search key < index/data record
1 !                           - 1,  search key > index/data record
1 !
1 ! SIDE EFFECTS:
1 !         RRV are skipped
1 !         AP is clobbered
1 !
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         COMMON_RAB_STR,
2         R_BKT_ADDR_STR,
2         R_REC_ADDR_STR,
2         R_IDX_DFN_STR;
2
2     LOCAL
2         BKTYP,                    ! Type of bucket we are searching
2         GOAL,                     ! Flag to say 'search past dups'
```

```
2            STATUS;                     ! Result of a given compare
2
2
2        ! If the record we are about to look at is an RRV, then we have an
2        ! empty bucket.
2        !
2        IF  .BKT_ADDR[BKT$B_LEVEL] EQL 0
2             AND
2            .BKT_ADDR[BKT$B_INDEXNO] EQLU 0
2        THEN
3            BEGIN
3            LOCAL
3                CNTRL: REF BBLOCK;  ! Control byte for 1st record
3
3            CNTRL = RM$CNTRL_ADDR(0);
3
3            IF .CNTRL [IRC$V_RRV]
3            THEN
4                BEGIN
4                IRAB [IRB$V_EMPT_SEEN] = 1;
4                RETURN GT
3                END;
2            END;
2
2        ! Should we be satisfied with an equal match or continue searching
2        ! for a greater than match ?
2        !
2        GOAL = EQ;
2
2        IF .IRAB [IRB$V_SRCHGT]
3            OR (.IRAB [IRB$V_POSINSERT] AND .BKT_ADDR [BKT$B_LEVEL] EQL 0)
2        THEN
2            GOAL = GT;
2
2        ! Now actually search the !!!!@@## bucket
2        ! First find out what kind of bucket it is
2        !
2
2        IF .BKT_ADDR [BKT$B_LEVEL] EQL 0
2        THEN
2            BKTYP = .IDX_DFN [IDX$B_DATBKTYP]
2        ELSE
2            BKTYP = .IDX_DFN [IDX$B_IDXBKTYP];
2
2        CASE .BKTYP FROM IDX$C_V2_BKT TO IDX$C_NCMPNCMP OF
2            SET
2            [IDX$C_V2_BKT]: ! Prologue two index bucket.
2                            !
2                            STATUS = SEARCH_V2 (.GOAL);
2
2            [IDX$C_CMPIDX]: ! Prologue three compressed index bucket.
2                            !
2                            STATUS = RM$SRCH_CMPR (.GOAL);
2
2            [IDX$C_NCMPIDX]: ! Prologue three noncompressed index bucket.
2                            !
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 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
```

```
1 ROUTINE RM$SEARCH_TREE : RL$RABREG_4567 =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine searches from the current record in the current bucket
1 !       to the stop level requested for a data record /index equal to or
1 !       greater than the search key.
1 !       NOTE: this routine should never be called by an outside routine
1 !             RM$CSEARCH_TREE should be used in its place
1 !
1 ! CALLING SEQUENCE:
1 !       RM$SEARCH_TREE()
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       BKT_ADDR                - address of current bucket
1 !       REC_ADDR                - address of record in bucket to start search at
1 !       IDX_DFN                 - address of index descriptor for current key of reference
1 !       IRAB                    - address of internal RAB
1 !       IRAB[KEYBUF2]           - address of search key
1 !       IRAB[IRB$B_KYSZ]        - size of key to compare(not equal to key size if generic search)
1 !       IRAB[IRB$B_STOPLEVEL]   - level to stop search at
1 !       IRAB[IRB$W_SRCHFLAGS]
1 !       IRAB[IRB$V_POSINSERT]   - if set, this is a position for insert
1 !       IRAB[IRB$V_POSDELETE]   - if set, this is position for delete
1 !       IRAB[IRB$V_SRCHGT]      - if set, this is a GT approximate search
1 !       IRAB[IRB$V_SRCHGE]      - if set, this is a GE approximate search
1 !       IRAB[IRB$V_FIRST_TIM]   - if set, this is the first seq. positioning
1 !                               - after a $connect or $rewind
1 !       IFAB[IFB$B_EXTRABUF]    - used to decide whether to try lockabove
1 !                               performance optimization coming down tree
1 !       IFAB[IFB$V_RU]          - if set, file is RU Journallable
1 !       IFAB[IFB$V_WRTACC]      - if set, file is write accessed
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       REC_ADDR                - address of index/data record which terminated search
1 !       IRAB[IRB$L_CURBDB]      - address of current BDB
1 !       BDB                     - address of current BDB
1 !       IRAB[IRB$L_LOCK_BDB]    - address of level above current if locked
1 !       IRAB[IRB$V_ABOVELCKD]   - set when level above data level locked
1 !       IRAB[IRB$V_DUPS_SEEN]   - set if non-deleted duplicate key encountered
1 !                                       at data level on position for insert
1 !       IRAB[IRB$V_DUP_KEY]     - set if duplicate key encountered at data level
1 !                                       on position for insert
1 !       IRAB [IRB$L_REC_COUNT]  - Set to the number of the record we found in
1 !                               a prologue three bucket
1 !
1 ! ROUTINE VALUE:
1 !       RMS$_RNF                - record not found
1 !       RMS$_SUC                - record found, in approximate search key
```

```
!                                     may not equal record/index key
!         RMS$_RLK                  - on an horizontal search at level zero for position
!                                     for insert, a lock error was encountered
!         miscellaneous I/O errors
!
! SIDE EFFECTS:
!         IRAB [ EMPT_SEEN ]        - may be clobbered at any level
!              [ EMPTY_BKT ]        - may be clobbered at data level
!
!--

    BEGIN

    LABEL
         BLK,
         BLK1,
         BLK2,
         BLK3,
         BLK4,
         BLK5,
         BLK6;

    BUILTIN
         AP,
         TESTBITSC;

    LOCAL
         VBN;

    EXTERNAL REGISTER
         COMMON_RAB_STR,
         COMMON_IO_STR,
         R_REC_ADDR_STR,
         R_IDX_DFN_STR;

    VBN = 0;

    DO
BLK2 :
         BEGIN

         LOCAL
              ST;

         ST = 0;

         ! if this is the 1st time, just get the 1st down pointer w/o searching
         !

         IF NOT .IRAB[IRB$V_FIRST_TIM]
         THEN
              ST = RM$SRCH_BY_KEY()
         ELSE
              IRAB [IRB$L_LST_NCMP] = .BKT_ADDR + BKT$C_OVERHDSZ;

         ! If the status is GT then no record in this bucket terminated the
         ! search. Therefore search horizontally for termination record.
```

```
                        ! Also search horiz when skipping over dups on position for insert
                        !
                        !

                        IF .ST GTR 0
                        THEN
                            BEGIN

                            LOCAL
                                SIZE;

                            IF .BKT_ADDR[BKT$V_LASTBKT]
                            THEN

                                IF .IRAB[IRB$V_POSINSERT]
                                THEN
                                    RETURN RMSSUC(SUC)
                                ELSE
                                    RETURN RMSERR(RNF);

                            BDB = .IRAB[IRB$L_CURBDB];
                            SIZE = .BDB[BDB$W_NUMB];

                            IF .VBN EQL 0
                            THEN
                                VBN = .BKT_ADDR[BKT$L_NXTBKT];

                            IF (.IRAB[IRB$V_POSINSERT] AND .BKT_ADDR[BKT$B_LEVEL] EQLU 0)
                            THEN
        BLK3 :                  BEGIN

                                ! Check if we can insert the new key into the left
                                !  hand bucket (the one we've got) or if we have to
                                !  go get the right hand bucket in order to make the
                                !  decision.
                                !
                                ! NEXT_DOWN will be equal to -1 if none was saved,
                                ! causing this test to fail.  Once this test has been
                                ! made once, NEXT_DOWN is set to zero, again causing
                                ! failure on this test.
                                !

                                IF .VBN EQLU .IRAB [IRB$L_NEXT_DOWN]
                                THEN
                                    ! Horizontal pointer = down pointer
                                    !

                                    IF  NOT (.IDX_DFN [IDX$V_DUPKEYS] AND .IDX_DFN [IDX$B_KEYREF] EQL 0)
                                    THEN
                                        ! Not a primary data bucket with possible duplicates
                                        !
                                        RETURN RMSSUC(SUC);

                                ! If the file is being shared in any way, and the bucket
                                ! in the level above was not locked coming down the tree,
                                ! we must re-access the level above to confirm that in
                                ! fact a split has not occurred and the level above index
```

```
        ! bucket doesn't have a new key value in the pointer to
        ! the bucket we are in.  If this is not done, the horizontal
        ! positioning logic that follows may cause the record to
        ! be inserted such that it is not accessible by random access.
        ! The following pictures illustrate.  Assume 1 record per
        ! bucket at the data level.
        !
        !   ----
        !  | 9  |  level 1    Index looks like this when coming
        !   ----              down tree looking to insert an "8".
        !    v
        !
        !  |-9--|  data level
        !
        !   db 1
        !
        !  |5 9 |         However, by the time that db 1 is actually
        !   ----          accessed, it has split and the index updated.
        !   /   \         Yet, because the key value "8" is less than the
        ! ---   ---       lowest key value in db 2, the position for
        !| 5 || 9 |       insert would be in db 1.  Rescanning the level
        ! ---   ---       1 bucket for key "8" will now find the correct
        !db 1  db 2       down pointer to db 2.
        !

        ! If the down pointer from level 1 can not have changed,
        ! either because the file is not shared or it is still
        ! locked, or it has already been checked once, then the
        ! check does not have to be made.
        !

        IF .IFAB [IFB$V_NORECLK]
            OR .IRAB [IRB$V_ABOVELCKD]
        THEN
            BEGIN
            IF .IRAB [IRB$L_NEXT_DOWN] NEQ 0
            THEN
                BEGIN
                IRAB [IRB$L_VBN_RIGHT] = 0;
                IRAB [IRB$L_NEXT_DOWN] = 0;
                IRAB [IRB$L_VBN_LEFT] = .BDB [BDB$L_VBN];
                END;

            LEAVE BLK3;
            END
        ELSE

            ! If NEXT_DOWN is zero, we've already been through
            ! here once, so don't check again.
            !

            IF .IRAB [IRB$L_NEXT_DOWN] EQL 0
            THEN
                LEAVE BLK3;

        ! The VBN of the level 1 bucket has been saved in the
```

```
		! VBN_RIGHT field.  Reaccess the bucket.  Release the
		! level 0 bucket and exit on errors.
		!

		ST = CACHE (.IRAB [IRB$L_VBN_RIGHT],
			.IDX_DFN [IDX$B_IDXBKTSZ]*512);
		IF NOT .ST
		THEN
		    BEGIN
		    BDB = .IRAB [IRB$L_CURBDB];
		    RM$RLSBKT(0);
		    IRAB [IRB$L_CURBDB] = 0;
		    RETURN .ST;
		    END;

		! Rescan the level 1 bucket just accessed.  If the
		! key value is no longer in the bucket at all, then
		! release both level 1 and level 0 buckets and come
		! down the tree from the top (returning RLK to CSEARCH_TREE
		! does that).
		!

		REC_ADDR = .BKT_ADDR + BKT$C_OVERHDSZ;
		IRAB [IRB$L_REC_COUNT] = 0;
		IF RM$SRCH_BY_KEY() GTR 0
		THEN
		    BEGIN
		    ST = RMSERR(RLK);
		    RM$RLSBKT(0);
		    BDB = .IRAB [IRB$L_CURBDB];
		    RM$RLSBKT(0);
		    IRAB [IRB$L_CURBDB] = 0;
		    RETURN .ST;
		    END;

		! At this point we've a down pointer in the same level
		! 1 bucket.  Check the pointer itself to see if it still
		! points to the level 0 bucket we found before.
		!

		BEGIN

		LOCAL
		    LEV0_BDB	: REF BBLOCK,
		    LEV1_VBN;

		LEV0_BDB = .IRAB [IRB$L_CURBDB];
		AP = 1;

		IF .IFAB [IFB$B_PLG_VER] LSSU PLG$C_VER_3
		THEN
		    LEV1_VBN = RM$RECORD_VBN()
		ELSE
		    LEV1_VBN = RM$V3_VBN();

		IF .LEV1_VBN NEQ  .LEV0_BDB [BDB$L_VBN]
		THEN
```

```
                        ! The pointer from the level 1 bucket is not the same,
                        ! but it is in the same level 1 bucket.  Simply release
                        ! the level 0 bucket we have accessed, and go around the
                        ! loop again so that this all happens again.
                        !
                        BEGIN
                        IRAB [IRB$B_SPL_BITS] = 0;
                        IRAB [IRB$L_CURBDB] = .BDB;
                        BDB = .LEV0_BDB;
                        RM$RLSBKT(0);
                        BDB = .IRAB [IRB$L_CURBDB];
                        BKT_ADDR = .BDB [BDB$L_ADDR];
                        REC_ADDR = .BKT_ADDR + BKT$C_OVERHDSZ;
                        LEAVE BLK2;
                        END;

                    ! Ok, everything is cool.  The down pointer is still the
                    ! same.  Release the level 1 bucket and prepare to follow
                    ! the horizontal links at the data level for primary key.
                    !

                    RM$RLSBKT(0);
                    BDB = .LEV0_BDB;
                    BKT_ADDR = .BDB [BDB$L_ADDR];
                    END;            ! of block defining LEV* locals.

                    IRAB [IRB$L_VBN_RIGHT] = 0;
                    IRAB [IRB$L_NEXT_DOWN] = 0;
                    IRAB [IRB$L_VBN_LEFT] = .BDB [BDB$L_VBN];

                    END;

                ! Under the following circumstances, the bucket to be read must
                ! be locked:
                !
                ! 1. If this is the stoplevel on any positioning for modification.
                ! 2. If this is a SIDR bucket and the file is write accessed.
                ! 3. If this is a primary data bucket, the file is write accessed
                !    and marked RU Journallable.
                !
                IF ((.IRAB[IRB$W_SRCHFLAGS]
                                AND
                                (IRB$M_POSINSERT + IRB$M_POSDELETE)) NEQ 0
                        AND
                        .IRAB[IRB$B_STOPLEVEL] EQL .BKT_ADDR[BKT$B_LEVEL])
                    OR
                  ((.IRAB[IRB$W_SRCHFLAGS]
                                AND
                                (IRB$M_POSINSERT + IRB$M_POSDELETE)) EQLU 0
                        AND
                        .IFAB[IFB$V_WRTACC]
                        AND
                        .BKT_ADDR[BKT$B_LEVEL] EQLU 0
                        AND
                        (.IFAB[IFB$V_RU]
                                OR
```

```
                        .IDX_DFN[IDX$B_KEYREF] GTRU 0))
        THEN
            IRAB [IRB$B_CACHEFLGS] = CSH$M_LOCK;

        ! Release current bucket before reading the next one.
        !

        RM$RLSBKT(0);
        IRAB [IRB$L_CURBDB] = 0;

        ! Read in the next bucket in the horizontal chain.
        !

        IF NOT (ST = RM$GETBKT(.VBN, .SIZE))
        THEN
            BEGIN

            ! If the level above was locked, release it.
            !

            IF TESTBITSC(IRAB[IRB$V_ABOVELCKD])
            THEN
                RELEASE(IRAB[IRB$L_LOCK_BDB]);

            RETURN .ST;
            END;

        VBN = .BKT_ADDR[BKT$L_NXTBKT];
        IRAB[IRB$L_CURBDB] = .BDB;

        IF (.IRAB[IRB$V_POSINSERT] AND .BKT_ADDR[BKT$B_LEVEL] EQLU 0)
        THEN

                ! This is a position for insert at level 0.
                !
                ! At this point we have just read in the next bucket in the
                ! horizontal chain.  The position for insert in the bucket
                ! just released was at the end of that bucket.
                !
                ! Basically, if the record is lower than any record in
                ! the next bucket, it should be inserted at the end of
                ! the previous bucket.
                !
                ! Field usage is as follows:
                !   IRAB[ VBN_LEFT ] - the leftmost, non-empty bucket in
                !         which the position for insert was at the end of
                !         the bucket.
                !
                !   IRAB[ VBN_RIGHT ] - the rightmost bucket in which the
                !         position for insert was at the beginning of the
                !         bucket, i.e., less than all existing values in
                !         that bucket, or an empty bucket.  This is used
                !         to determine if VBN_LEFT may be used after backing
                !         up.
                !
                ! Because the buckets are released prior to accessing the
                ! next bucket, it is possible for splits to occur prior to
```

```
                ! backing up, if that is necessary.  The following assumptions
                ! are made:
                ! 1) The position for insert can never be to the left of
                ! the initial VBN_LEFT.
                ! 2) The position for insert can never be in or to the right
                ! of a bucket in which the position for insert was at the
                ! beginning of that bucket.
                ! 3) If the NXTBKT link from a VBN_LEFT matches a previously
                ! accessed VBN_RIGHT, the correct position for insert is at
                ! the end of VBN_LEFT.
                ! 4) Empty buckets are skipped over until a non-empty bucket
                ! is encountered.
                !
                ! Note that an EMPT_SEEN bucket also returns greater than (GT)
                ! status.  VBN_LEFT and VBN_RIGHT are initialized to zero
                ! in the CSEARCH_TREE routine when positioning for insert and
                ! the stoplevel is 0.
                !

        WHILE 1 DO

        BEGIN

        REC_ADDR = .BKT_ADDR + BKT$C_OVERHDSZ;
        IRAB [IRB$L_REC_COUNT] = 0;

        ST = RM$SRCH_BY_KEY();

        IF (VBN=.BKT_ADDR[BKT$L_NXTBKT]) EQL .IRAB[IRB$L_VBN_RIGHT]
        THEN
            BEGIN
            IRAB [IRB$L_VBN_LEFT] = 0;
            IRAB [IRB$L_VBN_RIGHT] = 0;
            RETURN RMSSUC(SUC)
            END;

        IF .ST LSS 0
            AND
            .REC_ADDR EQLA (.BKT_ADDR + BKT$C_OVERHDSZ)
            OR
            (.BKT_ADDR[BKT$V_LASTBKT] AND .IRAB[IRB$V_EMPT_SEEN])
        THEN
            BEGIN

            IF .IRAB [IRB$L_VBN_RIGHT] EQL 0
            THEN
                IRAB [IRB$L_VBN_RIGHT] = .BDB [BDB$L_VBN];

            VBN = .IRAB [IRB$L_VBN_LEFT];
            END

        ELSE
            BEGIN

            IF .ST LEQ 0 OR .BKT_ADDR[BKT$V_LASTBKT]
            THEN
                BEGIN
```

```
7                               IRAB [IRB$L_VBN_LEFT] = 0;
7                               IRAB [IRB$L_VBN_RIGHT] = 0;
7                               RETURN RMSSUC(SUC);
6                               END;
6
6                       IF .IRAB [IRB$V_EMPT_SEEN]
6                       THEN
7                           BEGIN
7                           IF .IRAB [IRB$L_VBN_RIGHT] EQL 0
7                           THEN
7                               IRAB [IRB$L_VBN_RIGHT] = .BDB [BDB$L_VBN];
7                           END
6                       ELSE
7                           BEGIN
7                           IRAB [IRB$L_VBN_RIGHT] = 0;
7                           IRAB [IRB$L_VBN_LEFT] = .BDB [BDB$L_VBN];
6                           END;
5                       END;
5
5                   RM$RLSBKT(0);
5                   IRAB [IRB$L_CURBDB] = 0;
5                   IRAB [IRB$B_CACHEFLGS] = CSH$M_LOCK;
5
6                   IF NOT (ST=RM$GETBKT(.VBN,.SIZE))
5                   THEN
6                       BEGIN
6                       IF TESTBITSC (IRAB[IRB$V_ABOVELCKD])
6                       THEN
6                           RELEASE(IRAB[IRB$L_LOCK_BDB]);
6                       RETURN .ST
5                       END;
5
5                   IRAB [IRB$L_CURBDB] = .BDB;
5
4                   END;        ! of position for insert level 0 WHILE loop.
4               END             ! of greater than status from srch_by_key.
3           ELSE
3
3               ! status was less than or equal meaning we have a down pointer if
3               ! at an index level or may or may not have the data record
3               ! depending upon whether an exact match was desired or not
3               !
4               BEGIN
4
4               LOCAL
4                   SIZE;
4
4               IF .BKT_ADDR[BKT$B_LEVEL] EQLU .IRAB[IRB$B_STOPLEVEL]
4               THEN
5                   BEGIN
5
5                   IF .ST EQL 0
5                   THEN
6                       RETURN RMSSUC(SUC)
5                   ELSE
5
5                       IF .IRAB[IRB$V_POSINSERT]
```

```
5                       OR
5                       .IRAB[IRB$V_SRCHGT]
5                       OR
5                       .IRAB[IRB$V_SRCHGE]
5                   THEN
6                       RETURN RMSSUC(SUC)
5                   ELSE
5                       RETURN RMSERR(RNF);
5
4               END;
4
4           ! Get the down pointer from the index record positioned at.
4           !
4
4           IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
4           THEN
5               BEGIN
5               AP = 1;
5               VBN = RM$RECORD_VBN();
5               END
4           ELSE        ! Version 3 index bucket
5               BEGIN
5               VBN = RM$V3_VBN();
4               END;
4
4           IF .VBN EQLU 0
4           THEN
4               RETURN (RMSERR(TRE));
4
4           ! Check if about to read data level and switch to data level
4           ! transfer size.
4           !
4           IF .BKT_ADDR[BKT$B_LEVEL] EQLU 1
4           THEN
4               SIZE = .IDX_DFN[IDX$B_DATBKTSZ]
4           ELSE
4               SIZE = .IDX_DFN[IDX$B_IDXBKTSZ];
4
4           SIZE = .SIZE*512;
4
4           ! now check to see if the next level to read needs to be locked
4           !
4           !
4           BDB = .IRAB[IRB$L_CURBDB];          ! get curbdb into bdb
4
4           IF .IRAB[IRB$V_POSINSERT]
4               OR
4               .IRAB[IRB$V_POSDELETE]
4           THEN
4
4               ! lock the stop level if either position for insert or delete
4               !
4               !
4 BLK4 :
5               BEGIN
5
5               LOCAL
```

```
5                       STOPLEVCHK   : BYTE;
5
5               STOPLEVCHK = .IRAB[IRB$B_STOPLEVEL];
5               STOPLEVCHK = .STOPLEVCHK + 1;
5
5               IF .BKT_ADDR[BKT$B_LEVEL] EQL .STOPLEVCHK
5               THEN
5 BLK5 :
6                   BEGIN
6
6                   LOCAL
6                       REC_SIZE;
6
6                   ! if the stoplevel not zero, just lock it
6                   !
6
6                   IF .IRAB[IRB$B_STOPLEVEL] NEQ 0
6                       OR
6                       .IRAB[IRB$V_POSDELETE]
6                   THEN
7                       BEGIN
7                       IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
7                       LEAVE BLK5;
7
6                       END;
6
6                   ! try to save the next down pointer if it's in this bucket
6                   ! to avoid possible i/o at the data level
6                   !
7 BLK6 :            BEGIN
7
7                   IRAB [IRB$L_NEXT_DOWN] = -1;
7
7                   ! See if we're already at the end of the bucket.
7                   !
7                   CASE .IDX_DFN [IDX$B_IDXBKTYP] FROM IDX$C_V2_BKT TO IDX$C_NCMPIDX OF
7                   SET
8                   [IDX$C_V2_BKT]: BEGIN
8                                   REC_ADDR = .REC_ADDR + RM$REC_OVHD(1; REC_SIZE);
8                                   REC_ADDR = .REC_ADDR + .REC_SIZE;
7                                   END;
7
8                   [IDX$C_CMPIDX]: BEGIN
8                                   REC_ADDR = .REC_ADDR + .(.REC_ADDR)<0,8> + 2;
7                                   END;
7
8                   [IDX$C_NCMPIDX]:BEGIN
8                                   REC_ADDR = .REC_ADDR + .IDX_DFN [IDX$B_KEYSZ];
7                                   END;
7                   TES;
7
7                   IF .REC_ADDR GEQA .BKT_ADDR + .BKT_ADDR[BKT$W_FREESPACE]
7                   THEN
7                       LEAVE BLK6;
7
7                   IF .IDX_DFN [IDX$B_IDXBKTYP] EQL IDX$C_V2_BKT
7                   THEN
```

```
8                     BEGIN
8                     AP = 1;
8                     IRAB [IRB$L_NEXT_DOWN] = RM$RECORD_VBN();
8                     END
7                 ELSE
8                     BEGIN
8                     IRAB [IRB$L_REC_COUNT] = .IRAB [IRB$L_REC_COUNT] + 1;
8                     IRAB [IRB$L_NEXT_DOWN] = RM$V3_VBN();
7                     END;
7
6                 END;  ! of BLK6
6
6                 ! if we have extrabufs and this bucket is locked then
6                 ! we did lockabove optimization so set
6                 ! abovelckd to note that and save the bdb in lock_bdb.
6                 ! Also, if the blb_ptr is zero and there are extrabufs
6                 ! then the file is exclusively accessed therefore the
6                 ! optimization may also be made.
6                 !
6
6                 IF  .IRAB[IRB$B_BCNT] GTRU 2
6                 THEN
6                     IF .BDB[BDB$L_BLB_PTR] EQL 0
6                     THEN
7                         BEGIN
7                         IRAB[IRB$V_ABOVELCKD] = 1;
7                         IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK OR CSH$M_NOWAIT;
7                         IRAB[IRB$L_LOCK_BDB] = .BDB;
7                         IRAB[IRB$L_CURBDB] = 0;
7                         LEAVE BLK4;
7
7                         END
6                     ELSE
6                         IF .BBLOCK[.BDB[BDB$L_BLB_PTR], BLB$V_LOCK]
6                         THEN
7                             BEGIN
7                             IRAB[IRB$V_ABOVELCKD] = 1;
7                             IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK OR CSH$M_NOWAIT;
7                             IRAB[IRB$L_LOCK_BDB] = .BDB;
7                             IRAB[IRB$L_CURBDB] = 0;
7                             LEAVE BLK4;
7
6                             END;
6
6                 ! We are not keeping the level above locked on the
6                 ! way down.  Remember the vbn of this level 1 bucket we
6                 ! just passed through because it may have to be
6                 ! checked when we get to level 0.
6                 !
6
6                 IRAB [IRB$L_VBN_RIGHT] = .BDB [BDB$L_VBN];
6                 IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
6
5                 END
5             ELSE
5
5                 ! if position for insert and we have the extra buffers and
```

```
                    ! we are going all the way to level zero, then we will try
                    ! the performance optimization of keeping the level above
                    ! the data level locked to avoid going back to the root in
                    ! the event of a split.
                    !

                    IF .IRAB[IRB$V_POSINSERT]
                        AND
                        .IRAB[IRB$B_STOPLEVEL] EQL 0
                        AND
                        .IRAB[IRB$B_BCNT] GTRU 2
                    THEN
                        BEGIN
                        STOPLEVCHK = .STOPLEVCHK + 1;

                        IF .BKT_ADDR[BKT$B_LEVEL] EQL .STOPLEVCHK
                        THEN
                            BBLOCK[IRAB[IRB$B_CACHEFLGS], CSH$V_LOCK] = 1;

                        END;

                ! release the current bucket before going down
                !
                RM$RLSBKT(0);
                IRAB[IRB$L_CURBDB] = 0;
                END                                     ! of BLK4

            ! This is not a position for modification. Release the level above
            ! bucket, and if the next bucket to be obtained is a data bucket,
            ! the file is write accessed, and either the file is RU Journallable
            ! or the data bucket to be obtained is a SIDR bucket, then make
            ! sure the next bucket is locked.
            !
            ELSE
                BEGIN

                ! Determine whether the next bucket to be obtained will have to
                ! be locked.
                !
                IF .IFAB[IFB$V_WRTACC]
                    AND
                   .BKT_ADDR[BKT$B_LEVEL] EQLU 1
                    AND
                   (.IFAB[IFB$V_RU]
                        OR
                       .IDX_DFN[IDX$B_KEYREF] GTRU 0)
                THEN
                    IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;

                ! Release the current bucket.
                !
                RM$RLSBKT(0);
                IRAB[IRB$L_CURBDB] = 0;
                END;

            WHILE 1 DO
                BEGIN
```

```
5                   ST = RM$GETBKT(.VBN, .SIZE);
5
5                   ! if successful drop out of here
5                   !
5
5                   IF .ST
5                   THEN
5                       EXITLOOP;
5
5                   ! if aboveclkd then release it to
5                   ! avoid deadlock.  Save the VBN of the level 1
5                   ! bucket as it will need to be rechecked when level 0
5                   ! is reached.
5                   !
5                   IF TESTBITSC(IRAB[IRB$V_ABOVELCKD])
5                   THEN
6                       BEGIN
6                       BDB = .IRAB[IRB$L_LOCK_BDB];
6                       IRAB [IRB$L_VBN_RIGHT] = .BDB [BDB$L_VBN];
6                       IRAB[IRB$L_LOCK_BDB] = 0;
6                       RM$RLSBKT(0);
5                       END;
5
6                   IF .ST<0,16> NEQ RMSERR(RLK)
5                   THEN RETURN .ST;
5
5                   IRAB [IRB$B_CACHEFLGS] = CSH$M_LOCK;
4                   END;
4
4               IRAB[IRB$L_CURBDB] = .BDB;
4               VBN = 0;
3               END;                                    ! end else block
3
3           REC_ADDR = .BKT_ADDR + BKT$C_OVERHDSZ;
3           IRAB [IRB$L_REC_COUNT] = 0;
3           END
2       UNTIL 0                                         ! end until loop
1       END;
```

```
                                0C  BB 00000 RM$SEARCH_TREE:
                                                 PUSHR   #^M<R2,R3>                      ; 0892
               5E               0C  C2 00002     SUBL2   #12, SP                         ;
                          04    AE  D4 00005     CLRL    VBN                             ; 0984
                                6E  D4 00008 1$: 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                  ;
                      00000000G   00  16 00088         JSB     RM$CACHE                       ;
                 6E               50  D0 0008E         MOVL    R0, ST                         ;
                 1C               6E  E9 00091         BLBC    ST, 14$                        ; 1126
                 56          0E   A5  9E 00094         MOVAB   14(R5), REC_ADDR               ; 1142
                           0094   C9  D4 00098         CLRL    148(IRAB)                      ; 1143
                           FEF6   30     0009C         BSBW    RM$SRCH_BY_KEY                 ; 1144
                                  50  D5 0009F         TSTL    R0                             ;
                                  1F  15 000A1         BLEQ    15$                            ;
                 6E        82AA   8F  3C 000A3         MOVZWL  #33450, ST                     ; 1147
                                  7E  D4 000A8         CLRL    -(SP)                          ; 1148
                          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  D0  000E1         MOVL    BDB, 32(IRAB)                     ; 1185
                     54             52  D0  000E5         MOVL    LEV0_BDB, BDB                     ; 1186
                                    7E  D4  000E8         CLRL    -(SP)                             ; 1187
                                 0000G  30  000EA         BSBW    RM$RLSBKT                         ;
                     5E             04  C0  000ED         ADDL2   #4, SP                            ;
                     54       20    A9  D0  000F0         MOVL    32(IRAB), BDB                     ; 1188
                     55       18    A4  D0  000F4         MOVL    24(BDB), BKT_ADDR                 ; 1189
                     56       0E    A5  9E  000F8         MOVAB   14(R5), REC_ADDR                  ; 1190
                                  FF09  31  000FC         BRW     1$                                ; 1191
                                    7E  D4  000FF  18$:   CLRL    -(SP)                             ; 1199
                                 0000G  30  00101         BSBW    RM$RLSBKT                         ;
                     5E             04  C0  00104         ADDL2   #4, SP                            ;
                     54             52  D0  00107         MOVL    LEV0_BDB, BDB                     ; 1200
                     55       18    A4  D0  0010A         MOVL    24(BDB), BKT_ADDR                 ; 1201
                            008C    C9  7C  0010E  19$:   CLRQ    140(IRAB)                         ; 1204
           0088      C9       1C    A4  D0  00112         MOVL    28(BDB), 136(IRAB)                ; 1206
                     05       42    A9  93  00118  20$:   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  D0  0015A         MOVL    R0, ST                            ;
                     03             6E  E8  0015D         BLBS    ST, 24$                           ;
                                  00A1  31  00160         BRW     36$                               ;
             04      AE       08    A5  D0  00163  24$:   MOVL    8(BKT_ADDR), VBN                  ; 1261
             20      A9             54  D0  00168         MOVL    BDB, 32(IRAB)                     ; 1262
                     03       42    A9  E8  0016C         BLBS    66(IRAB), 26$                     ; 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  D0  00183         MOVL    R0, ST                            ;
             04      AE       08    A5  D0  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 0035B          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  05 003BA          RSB                                            ;
```

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

; 1612          1672  1
```

```
1 GLOBAL ROUTINE RM$CSEARCH_TREE : RL$RABREG_67 =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This module controls the search through a tree.  If a start position
1 !       is given, the search continues from that point.  If no start position is
1 !       given, then the search begins at the root.  In the position for insert case,
1 !       when a bucket is found on a horizontal search, a record lock error is
1 !       returned from the search tree routine.  This controlling module then
1 !       restarts the search from the root to prevent deadlock.
1 !
1 ! CALLING SEQUENCE:
1 !       RM$CSEARCH_TREE()
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       IRAB                    - address of internal RAB
1 !       IRAB[KEYBUFFER 2]       - address of search key
1 !       IRAB[IRB$B_KYSZ]        - size of key to compare(not equal to total key size if
1 !                                 generic search
1 !       IRAB[IRB$B_STOPLEVEL]   - level to stop search at
1 !       IRAB[IRB$W_SRCHFLAGS]   -
1 !       IRAB[IRB$V_POSINSERT]   - if set, this is a position for insert
1 !       IRAB[IRB$V_SRCHGT]      - if set, this is a GT approximate search
1 !       IRAB[IRB$V_SRCHGE]      - if set, this is a GE approximate search
1 !       IRAB[IRB$V_FIRST_TIM]   - if set, this is the 1ST sequential positioning
1 !                               - after a $connect or $rewind
1 !       IRAB[IRB$V_NORLS_RNF]   - if set, do not release on RNF error
1 !       IRAB[IRB$L_CURBDB]      - if zero, then start at root
1 !       IRAB[IRB$L_CURBDB]      - if non-zero, then
1 !       REC_ADDR                - address of record in bucket to start search at
1 !       IDX_DFN                 - address of current index descriptor
1 !
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       REC_ADDR                - address of index/data record which terminated search
1 !       IRAB[IRB$L_CURBDB]      - address of current BDB
1 !       IRAB[IRB$L_LOCK_BDB]    - address of level above current if locked
1 !       IRAB[IRB$V_ABOVELCKD]   - set if level above data level locked
1 !       IRAB[IRB$V_NORLS_RNF]   - always cleared
1 !
1 ! ROUTINE VALUE:
1 !       RMS$_RNF                - record not found
1 !       RMS$_SUC                - record found, in approximate search key
1 !                                 may not equal record/index key
1 !       Miscellaneous I/O errors
1 !
1 ! SIDE EFFECTS:
1 !       if successful, bucket is accessed
1 !       if RNF and NORLS_RNF is set, then bucket is accessed
```

```
  1 !       bucket released on all other errors
  1 !       permanence is set in the root's bdb, if not pos-for-insert
  1 !               or there are extra buffers hanging around
  1 !
  1 !--
  1
  2     BEGIN
  2
  2     EXTERNAL REGISTER
  2         COMMON_RAB_STR,
  2         R_REC_ADDR_STR,
  2         R_IDX_DFN_STR;
  2
  2     GLOBAL REGISTER
  2         R_BDB_STR,
  2         R_BKT_ADDR_STR;
  2
  2     IRAB[IRB$V_ABOVELCKD] = 0;              ! make sure this is clear at the start
  2
  2     WHILE 1
  2     DO
  3         BEGIN
  3         BDB = .IRAB[IRB$L_CURBDB];
  3
  3         IF .BDB EQLU 0
  3         THEN
  3
  3             ! If no position given, start at root
  3             !
  4             BEGIN
  4
  4             ! If the index has not been initialized, read in the index des
  4             ! criptor again if it still has not been initialized, return error
  4             ! Once the index descriptors are shared, this code can come out
  4             ! since when the index is made, the in_core index descriptors will
  4             ! be updated.
  4             !
  4
  4             IF .IDX_DFN[IDX$V_INITIDX]
  4             THEN
  5                 BEGIN
  5                 IRAB[IRB$V_NEW_IDX] = 1;
  5
P 5                 RETURN_ON_ERROR (RM$KEY_DESC(.IDX_DFN[IDX$B_KEYREF]),
P 5                     BEGIN
P 5                     (IRAB[IRB$V_NORLS_RNF] = 0)
  5                     END);
  5
  5                 IF .IDX_DFN[IDX$V_INITIDX]
  5                 THEN
  6                     (IRAB[IRB$V_NORLS_RNF] = 0;
  6
  6                     IF .IRAB[IRB$V_POSINSERT]
  6                     THEN
  7                         RETURN RMSERR(IDX)
  6                     ELSE
  5                         RETURN RMSERR(RNF));
```

```
	END;

    ! Read root and make sure it is still the root.  If not reread
    ! prologue to obtain root VBN.
    !

    WHILE 1
    DO
	BEGIN

	! if pos for insert then -- if the root should turn out to be
	! the stoplevel then lock it
	!

	IF .IRAB[IRB$V_POSINSERT]
	THEN
	    BEGIN

	    IF .IRAB[IRB$B_STOPLEVEL] EQL 0
	    THEN
		BEGIN

		IRAB [IRB$L_VBN_LEFT] = 0;
		IRAB [IRB$L_VBN_RIGHT] = 0;
		IRAB [IRB$L_NEXT_DOWN] = 0;

		! If the root is at level 1, we want to save
		! it's VBN so that the down pointer from it can
		! be checked when level 0 is reached.
		!

		IF .IDX_DFN [IDX$B_ROOTLEV] EQL 1
		THEN
		    IRAB [IRB$L_VBN_RIGHT] = .IDX_DFN [IDX$L_ROOTVBN];
		END
	    ELSE

		IF .IDX_DFN[IDX$B_ROOTLEV] EQL .IRAB[IRB$B_STOPLEVEL]
		THEN
		    IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;

	    END;

	! try to get the root bucket
	!

	RETURN_ON_ERROR (RM$GETBKT(.IDX_DFN[IDX$L_ROOTVBN],
		.IDX_DFN[IDX$B_IDXBKTSZ]*512),
	    BEGIN
	    (IRAB[IRB$V_NORLS_RNF] = 0)
	    END);

	! if it really is the root bucket, leave this loop
	!

	IF .BBLOCK[.BDB[BDB$L_ADDR], BKT$V_ROOTBKT]
```

```
                THEN
                    EXITLOOP;

                RM$RLSBKT(0);
                IRAB[IRB$V_NEW_IDX] = 1;

                RETURN_ON_ERROR (RM$KEY_DESC(.IDX_DFN[IDX$B_KEYREF]),
                    BEGIN
                    (IRAB[IRB$V_NORLS_RNF] = 0)
                    END);

                END;

            IRAB[IRB$L_CURBDB] = .BDB;

            IF NOT .IRAB[IRB$V_POSINSERT]
                OR
                (.IDX_DFN[IDX$B_KEYREF] LSSU .IFAB[IFB$B_EXTRABUF])
            THEN
                BDB[BDB$V_PRM] = 1;

            REC_ADDR = .BDB[BDB$L_ADDR] + BKT$C_OVERHDSZ;
            IRAB[IRB$L_REC_COUNT] = 0;
            END
        ELSE
            BEGIN

            IF .IRAB[IRB$V_FIRST_TIM]
            THEN
                RETURN RMSERR (BUG);

            BKT_ADDR = .BDB[BDB$L_ADDR]
            END;

        BEGIN

        LOCAL
            STS;

        STS = RM$SEARCH_TREE();

        IF .STS<0, 16> NEQU RMSERR(RLK)
        THEN
            BEGIN

            IF .STS<0, 16> EQL RMSERR(RNF)
                AND
                NOT .IRAB[IRB$V_NORLS_RNF]
            THEN
                BEGIN
                IRAB[IRB$L_CURBDB] = 0;
                RM$RLSBKT(0)
                END;

            IRAB[IRB$V_NORLS_RNF] = 0;

            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$:  MOVL    24(BDB), BKT_ADDR               ; 1875
                             FB70  30 000D2  14$:  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 000F0        BSBW    RM$RLSBKT                       ;
                    5E         04  C0 000F3        ADDL2   #4, SP                          ;
             42     A9         20  8A 000F6  15$:  BICB2   #32, 66(IRAB)                   ; 1898
                    50         51  D0 000FA        MOVL    STS, R0                         ; 1900
                               06  11 000FD        BRB     17$                             ;
                          44   A9  94 000FF  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
```

0328 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RM3SSIDR
LIS

RM3UPSIDX
LIS

RM3SRCHKY
LIS

RM3UPDDEL
LIS

RM3UPDATE
LIS