```
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
```

```
_$
Sy
---
NT$
NT$
NT$
NT$
NT$
NT$

NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$

NT$

NT'
NT$
NT$
NT$
NT$
NT'
NT

NT
NT
NT
NT
NT
NT
PI
```

```
                                   B 13
**FILE**ID**RM3MAKIDX









RRRRRRRR    MM      MM    333333    MM      MM    AAAAAA    KK      KK    IIIIII    DDDDDDDD    XX      XX
RRRRRRRR    MM      MM    333333    MM      MM    AAAAAA    KK      KK    IIIIII    DDDDDDDD    XX      XX
RR      RR  MMMM  MMMM  33      33  MMMM  MMMM  AA      AA  KK      KK      II      DD      DD  XX      XX
RR      RR  MMMM  MMMM  33      33  MMMM  MMMM  AA      AA  KK      KK      II      DD      DD  XX      XX
RR      RR  MM  MM  MM          33  MM  MM  MM  AA      AA  KK    KK        II      DD      DD    XX  XX
RR      RR  MM  MM  MM          33  MM  MM  MM  AA      AA  KK    KK        II      DD      DD    XX  XX
RRRRRRRR    MM      MM        33    MM      MM  AA      AA  KKKKKK          II      DD      DD      XX
RRRRRRRR    MM      MM        33    MM      MM  AA      AA  KKKKKK          II      DD      DD      XX
RR  RR      MM      MM          33  MM      MM  AAAAAAAAAA  KK    KK        II      DD      DD    XX  XX
RR  RR      MM      MM          33  MM      MM  AAAAAAAAAA  KK    KK        II      DD      DD    XX  XX
RR    RR    MM      MM  33      33  MM      MM  AA      AA  KK      KK      II      DD      DD  XX      XX      ....
RR    RR    MM      MM  33      33  MM      MM  AA      AA  KK      KK      II      DD      DD  XX      XX      ....
RR      RR  MM      MM    333333    MM      MM  AA      AA  KK      KK    IIIIII    DDDDDDDD    XX      XX      ....
RR      RR  MM      MM    333333    MM      MM  AA      AA  KK      KK    IIIIII    DDDDDDDD    XX      XX      ....


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

RM3M
V04-

```
0 MODULE RM3MAKIDX (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 !*  CORPORA*ION.                                                            *
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 makes an index given a key of reference
1 !
1 !
1 ! ENVIRONMENT:
1 !
1 !              VAX/VMS OPERATING SYSTEM
1 !
1 !--
1
1 !
1 ! AUTHOR:        D. H. Gillespie            CREATION DATE:            2-AUG-78  8:51
1 !
1 !
1 !
1 ! MODIFIED BY:
1 !
1 !   V03-008    DAS0001        David Solomon          25-Mar-1984
1 !              Fix broken branches. Make RM$MAKE_HIGH_KY not a global routine.
1 !
1 !   V03-007    MCN0003        Maria del C. Nasr      31-Mar-1983
1 !              More linkages reorganization
1 !
1 !   V03-006    MCN0002        Maria del C. Nasr      28-Feb-1982
1 !              Reorganize linkages
```

```
1 !
1 !     V03-005      DAS0001          David Solomon           28-Jan-1983
1 !                  Add support for 64-bi. binary keys to RM$MAKE_HIGH_KY.
1 !
1 !     V03-004      MCN0001          Maria del C. Nasr       29-Oct-1982
1 !                  Call for RM$MAKE_HIGH_KY for prologue 3 non-compressed
1 !                  keys so that the indexed is formatted depending on the
1 !                  key data type.
1 !
1 !     V03-003      TMK0002          Todd M. Katz            11-Sep-1982
1 !                  Eliminate the linkage for RM$ADD_TO_ARRAY which is not called
1 !                  within this module.
1 !
1 !     V03-002      KBT0169          Keith B. Thompson       23-Aug-1982
1 !                  Reorganize psects
1 !
1 !     V03-001      KBT0062          Keith B. Thompson       11-Jun-1982
1 !                  Get rid of the index descriptor offset calculation
1 !
1 !     V02-C        TMK0001          Todd M. Katz            01-Mar-1982
1 !                  Add support for rear end truncation of keys in the index
1 !                  of prolog 3 files with compressed indicies. The change
1 !                  made is to RM$MAK_IDX_REC. The high key need only contain
1 !                  one FF!
1 !
1 !     V02-006      PSK0003          Paulina S. Knibbe       09-Aug-1981
1 !                  Make RM$MAK_IDX_REC into a global routine so NEW_ROOT
1 !                  can call it.
1 !
1 !     V02-005      PSK0002          Paulina S. Knibbe       02-Aug-1981
1 !                  Remove support for rear-end truncation of keys in index
1 !
1 !     V02-004      PSK0001          Paulina S. Knibbe       29-May-1981
1 !                  Add support for making prologue three indexes
1 !
1 !     V02-003      REFORMAT         Paulina S. Knibbe       23-Jul-1980
1 !
1 ! REVISION HISTORY:
1 !
1 !     Wendy Koenig,        24-OCT-78  14:02
1 !     X0002 - MAKE CHANGES CAUSED BY SHARING CONVENTIONS
1 !
1 !*****
1
1 LIBRARY 'RMSLIB:RMS';
1
1 REQUIRE 'RMSSRC:RMSIDXDEF';
1
1 ! define default psects for code
1 !
1 PSECT
1     CODE = RM$RMS3(PSECT_ATTR),
1     PLIT = RM$RMS3(PSECT_ATTR);
1
1 ! Linkages
1 !
1 LINKAGE
```

```
;  115          0179  1         L_CHKSUM,
;  116          0180  1         L_PRESERVE1,
;  117          0181  1         L_RABREG_67,
;  118          0182  1         L_RABREG_7,
;  119          0183  1         L_RELEASE;
;  120          0184  1
;  121          0185  1 LINKAGE
;  122          0186  1         RL$RELEASE_KD           = JSB () : GLOBAL (COMMON_RABREG);
;  123          0187  1
;  124          0188  1 !
;  125          0189  1 ! Forward Routines
;  126          0190  1 !
;  127          0191  1 FORWARD ROUTINE
;  128          0192  1         MAKE_HIGH_KY            : RL$RABREG_67 NOVALUE,
;  129          0193  1         RELEASE_KEYDESC         : RL$RELEASE_KD NOVALUE,
;  130          0194  1         RM$MAKE_INDEX           : RL$RABREG_7,
;  131          0195  1         RM$MAK_IDX_REC          : RL$RABREG_67 NOVALUE;
;  132          0196  1
;  133          0197  1 ! External Routines
;  134          0198  1 !
;  135          0199  1 EXTERNAL ROUTINE
;  136          0200  1         RM$AL_FRMT_BKT          : RL$RABREG_7,
;  137          0201  1         RM$KEY_DESC             : RL$RABREG_7,
;  138          0202  1         RM$MAKSUM               : RL$CHKSUM,
;  139          0203  1         RM$RELEASE              : RL$RELEASE ADDRESSING_MODE( LONG_RELATIVE ),
;  140          0204  1         RM$VBN_SIZE             : RL$PRESERVE1;
;  141          0205  1
```

```
1 %SBTTL  'MAKE_HIGH_KY'
1 ROUTINE MAKE_HIGH_KY : RL$RABREG_67 NOVALUE =
1
1 !++
1 !
1 ! MAKE_HIGH_KY -
1 !
1 !       This routine formats a high key depending on the key type
1 !       at REC_ADDR and returns REC_ADDR beyond high key.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       MAKE_H.GH_KY()
1 !
1 ! INPUT PARAMETERS:
1 !       none
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       REC_ADDR        - record pointer
1 !       IDX_DFN         - address of index descriptor
1 !
1 ! OUTPUT PARAMETERS:
1 !       none
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       REC_ADDR        - updated to point beyond  high key
1 !
1 ! ROUTINE VALUE:
1 !       none
1 !
1 ! SIDE EFFECTS:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         R_REC_ADDR,
2         R_IDX_DFN_STR;
2
2     ! If the data type is anything but packed decimal, set high key to 255.
2     ! Then if type is signed binary , clear sign bit.
2     !
2
2     IF .IDX_DFN[IDX$B_DATATYPE] NEQU IDX$C_PACKED
2     THEN
3         BEGIN
3         REC_ADDR = CH$FILL(%X'FF', .IDX_DFN[IDX$B_KEYSZ], .REC_ADDR);
3
4         IF ( .IDX_DFN[IDX$B_DATATYPE] EQL IDX$C_SGNWORD )
3             OR
4             ( .IDX_DFN[IDX$B_DATATYPE] EQL IDX$C_SGNLONG )
3             OR
4             ( .IDX_DFN[IDX$B_DATATYPE] EQL IDX$C_SGNQUAD )
```

```
;  200          0263  3          THEN
;  201          0264  3              (.REC_ADDR - 1)<0, 8> = %X'7F';
;  202          0265  3
;  203          0266  3          END
;  204          0267  2      ELSE
;  205          0268  2
;  206          0269  2          ! When the key is packed decimal, fill nibbles with '9's except for
;  207          0270  2          ! size which if 'C'.
;  208          0271  2          !
;  209          0272  3          BEGIN
;  210          0273  3          REC_ADDR = CH$FILL(%X'99', .IDX_DFN[IDX$B_KEYSZ] - 1, .REC_ADDR);
;  211          0274  3          (.REC_ADDR)<0, 8> = %X'9C';
;  212          0275  3          REC_ADDR = .REC_ADDR + 1;
;  213          0276  2          END;
;  214          0277  2
;  215          0278  1      END;


                                                                       .TITLE  RM3MAKIDX
                                                                       .IDENT  \V04-000\

                                                                       .EXTRN  RM$AL_FRMT_BKT, RM$KEY_DESC
                                                                       .EXTRN  RM$MAKSUM, RM$RELEASE
                                                                       .EXTRN  RM$VBN_SIZE

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

                                                     3C  BB 00000 MAKE_HIGH_KY:
                                                                       PUSHR   #^M<R2,R3,R4,R5>                          ; 0207
                                     05         1D   A7  91 00002      CMPB    29(IDX_DFN), #5                           ; 0253
                                                     27  13 00006      BEQL    2$
                                     50         20   A7  9A 00008      MOVZBL  32(IDX_DFN), R0                           ; 0256
           50        FF    8F        6E              00  2C 0000C      MOVC5   #0, (SP), #255, R0, (REC_ADDR)            ;
                                                     66     00012                                                        ;
                                     56              53  D0 00013      MOVL    R3, REC_ADDR                              ;
                                     01         1D   A7  91 00016      CMPB    29(IDX_DFN), #1                           ; 0258
                                                     0C  13 0001A      BEQL    1$
                                     03         1D   A7  91 0001C      CMPB    29(IDX_DFN), #3                           ; 0260
                                                     06  13 00020      BEQL    1$
                                     06         1D   A7  91 00022      CMPB    29(IDX_DFN), #6                           ; 0262
                                                     1B  12 00026      BNEQ    3$
                               FF    A6         7F   8F  90 00028 1$:  MOVB    #127, -1(REC_ADDR)                        ; 0264
                                                     14  11 0002D      BRB     3$                                        ; 0253
                                     50         20   A7  9A 0002F 2$:  MOVZBL  32(IDX_DFN), R0                           ; 0273
                                                     50  D7 00033      DECL    R0
           50        99    8F        6E              00  2C 00035      MOVC5   #0, (SP), #153, R0, (REC_ADDR)            ;
                                                     66     0003B                                                        ;
                                     56              53  D0 0003C      MOVL    R3, REC_ADDR                              ;
                                     86         9C   8F  90 0003F      MOVB    #-100, (REC_ADDR)+                        ; 0274
                                                     3C  BA 00043 3$:  POPR    #^M<R2,R3,R4,R5>                          ; 0278
                                                     05     00045      RSB                                               ;

; Routine Size:  70 bytes,    Routine Base:  RM$RMS3 + 0000


;  216          0279  1
```

```
:   2'8        0280  1 %SBTTL  'RELEASE_KEYDESC'
:   219        0281  1 ROUTINE RELEASE_KEYDESC : RL$RELEASE_KD NOVALUE =
:   220        0282  1
:   221        0283  1 !++
:   222        0284  1 !
:   223        0285  1 ! RELEASE_KEYDESC
:   224        0286  1 !
:   225        0287  1 !       This routine releases the locked key descriptor whose BDB is stored
:   226        0288  1 !       in IRAB[IRB$L_LOCK_BDB].
:   227        0289  1 !
:   228        0290  1 ! CALLING SEQUENCE:
:   229        0291  1 !       RELEASE_KEYDESC()
:   230        0292  1 !
:   231        0293  1 ! INPUT PARAMETERS:
:   232        0294  1 !       NONE
:   233        0295  1 !
:   234        0296  1 ! IMPLICIT INPUTS:
:   235        0297  1 !       IRAB                 - address of internal RAB structure
:   236        0298  1 !       IRAB[IRB$L_LOCK_BDB]     - BDB of key descriptor buffer
:   237        0299  1 !
:   238        0300  1 ! OUTPUT PARAMETERS:
:   239        0301  1 !       NONE
:   240        0302  1 !
:   241        0303  1 ! IMPLICIT OUTPUTS:
:   242        0304  1 !       NONE
:   243        0305  1 !
:   244        0306  1 ! ROUTINE VALUE:
:   245        0307  1 !       NONE
:   246        0308  1 !
:   247        0309  1 ! SIDE EFFECTS:
:   248        0310  1 !       The lock on the key descriptor is released.
:   249        0311  1 !       IRAB[IRB$L_LOCK_BDB] = 0
:   250        0312  1 !
:   251        0313  1 !--
:   252        0314  1
:   253        0315  2     BEGIN
:   254        0316  2
:   255        0317  2     EXTERNAL REGISTER
:   256        0318  2         COMMON_RAB_STR;
:   257        0319  2
:   258        0320  2     GLOBAL REGISTER
:   259        0321  2         R_BDB_STR;
:   260        0322  2
:   261        0323  2     BDB = .IRAB[IRB$L_LOCK_BDB];
:   262        0324  2     IRAB[IRB$L_LOCK_BDB] = 0;
:   263        0325  2     RM$RELEASE(0);
:   264        0326  1     END;


                                               1C  BB 00000 RELEASE_KEYDESC:
                                                                   PUSHR   #^M<R2,R3,R4>                          : 0281
                                54       0084  C9  D0 00002        MOVL    132(IRAB), BDB                         : 0323
                                         0084  C9  D4 00007        CLRL    132(IRAB)                              : 0324
                                               53  D4 0000B        CLRL    R3                                     : 0325
```

```
                               00000000G  EF  16 0000D          JSB     RM$RELEASE                       ;
                                          1C  BA 00013          POPR    #^M<R2,R3,R4>                    ; 0326
                                              05 00015          RSB                                      ;

; Routine Size:  22 bytes,    Routine Base:  RM$RMS3 + 0046


;  265          0327  1
;   267         0328  1 %SBTTL  'RM$MAKE_INDEX'
;   268         0329  1 GLOBAL ROUTINE RM$MAKE_INDEX : RL$RABREG_7 =
;   269         0330  1
;   270         0331  1 !++
;   271         0332  1 !
;   272         0333  1 ! RM$MAKE_INDEX - This routine builds an index for the given key of reference
;   273         0334  1 !
;   274         0335  1 ! CALLING SEQUENCE:
;   275         0336  1 !       RM$MAKE_INDEX()
;   276         0337  1 !
;   277         0338  1 ! INPUT PARAMETERS:
;   278         0339  1 !       NONE
;   279         0340  1 !
;   280         0341  1 ! IMPLICIT INPUTS:
;   281         0342  1 !
;   282         0343  1 !       IDX_DFN         - address of in core key descriptor which needs an index
;   283         0344  1 !       IRAB            - address of internal RAB
;   284         0345  1 !
;   285         0346  1 . OUTPUT PARAMETERS:
;   286         0347  1 !       NONE
;   287         0348  1 !
;   288         0349  1 ! IMPLICIT OUTPUTS:
;   289         0350  1 !       NONE
;   290         0351  1 !
;   291         0352  1 ! ROUTINE VALUE:
;   292         0353  1 !       NONE
;   293         0354  1 !
;   294         0355  1 ! SIDE EFFECTS:
;   295         0356  1 !
;   296         0357  1 !       The index is made if necessary with disk and in_core key descriptors
;   297         0358  1 !       being updated.  All IRAB BDB's are used but zeroed once  descriptors are
;   298         0359  1 !       released.  IRAB [ IRB$L_MIDX_TMPX ] 's are used as scratch areas.
;   299         0360  1 !
;   300         0361  1 !
;   301         0362  1 !--
;   302         0363  1
;   303         0364  2     BEGIN
;   304         0365  2
;   305         0366  2     MACRO
;   306         0367  2         LOWER_VBN = IRAB [ IRB$L_MIDX_TMP1 ]%,
;   307         0368  2         LEVEL = IRAB [ IRB$L_MIDX_TMP2 ]%;
;   308         0369  2
;   309         0370  2     EXTERNAL REGISTER
;   310         0371  2         COMMON_RAB_STR,
;   311         0372  2         R_IDX_DFN_STR;
;   312         0373  2
;   313         0374  2     ! There should be nothing locked when an index is made for the primary key.
;   314         0375  2     ! Only a record lock exists when the secondary index is made. Lock the disk
;   315         0376  2     ! key descriptor storing it's BDB in IRAB [ IRB$L_LOCK_BDB ]. Check that the
;   316         0377  2     ! index has not been made.  If it has, release lock and and return.
;   317         0378  2     ! In_core descriptor has been updated by read and lock. If the index has
;   318         0379  2     ! not been made, precede to build it.
;   319         0380  2     !
;   320         0381  2     ! Force new read of key descriptor
;   321         0382  2     !
;   322         0383  2     IRAB [ IRB$V_NEW_IDX ] = 1;
;   323         0384  2
;   324          0385  2        ! Lock descriptor so no one else can monkey
;   325          0386  2        !
;   326          0387  2        IRAB [ IRB$B_CACHEFLGS ] = CSH$M_LOCK;
;   327          0388  2
;   328          0389  2        RETURN_ON_ERROR( RM$KEY_DESC( .IDX_DFN [ IDX$B_KEYREF] ) );
;   329          0390  2
;   330          0391  2        ! double check that no one else has made index.
;   331          0392  2        !
;   332          0393  3        BEGIN
;   333          0394  3
;   334          0395  3        GLOBAL REGISTER
;   335          0396  3            R_BDB_STR;
;   336          0397  3
;   337          0398  3        BDB = .IRAB [ IRB$L_LOCK_BDB ];
;   338          0399  3
;   339          0400  3        IF NOT .IDX_DFN [ IDX$V_INITIDX ]
;   340          0401  3        THEN
;   341          0402  4            BEGIN
;   342          0403  4            RELEASE_KEYDESC();
;   343          0404  4            RETURN T
;   344          0405  4
;   345          0406  3            END;
;   346          0407  3
;   347          0408  3        ! Point to the descriptor in the block
;   348          0409  3        !
;   349          0410  3        IRAB [ IRB$L_MIDX_TMP3 ] = .BDB [ BDB$L_ADDR ] + .IDX_DFN [ IDX$W_OFFSET ];
;   350          0411  3
;   351          0412  3        ! Invalidate buffer so no need to back out
;   352          0413  3        !
;   353          0414  3        BDB [ BDB$V_VAL ] = 0
;   354          0415  3
;   355          0416  2        END;         ! End global definition of COMMON_IO_STR
;   356          0417  2
;   357          0418  2        ! It is necessary to build the index.  Start with tne data level and work
;   358          0419  2        ! up to the root, taking care to have 2 levels of index if LANUN is not
;   359          0420  2        ! equal IANUM.
;   360          0421  2        !
;   361        P 0422  2        RETURN_ON_ERROR( RM$AL_FRMT_BKT( .IDX_DFN [ IDX$B_DANUM ],
;   362        P 0423  2                                         .IDX_DFN [ IDX$B_DATBKTSZ ] * 512 ),
;   363        P 0424  2                         BEGIN
;   364        P 0425  2                         RELEASE_KEYDESC()
;   365          0426  2                         END );
;   366          0427  2
;   367          0428  2        ! Finish formatting data level bucket.
;   368          0429  2        !
;   369          0430  3        BEGIN
;   370          0431  3
;   371          0432  3        LOCAL
;   372          0433  3            BDB     : REF BBLOCK,
;   373          0434  3            BUCKET  : REF BBLOCK,
;   374          0435  3            VBN;
;   375          0436  3
;   376          0437  3        BDB = .IRAB [ IRB$L_NXTBDB ];                ! BDB of data bucket
;   377          0438  3        BUCKET = .BDB [ BDB$L_ADDR ];                ! address of data bucket
;   378          0439  3        VBN = .BDB [ BDB$L_VBN ];
;   379          0440  3        BUCKET [ BKT$L_NXTBKT ] = .VBN;              ! forward link is self
;   380          0441  3
;  381        0442  3      ! Save first data bucket VBN in disk key descriptor
;  382        0443  3      !
;  383        0444  3      BBLOCK [ .IRAB [ IRB$L_MIDX_TMP3 ],KEY$L_LDVBN ] = .VBN;
;  384        0445  3      BUCKET [ BKT$B_LEVEL ] = 0;
;  385        0446  3      BUCKET [ BKT$B_BKTCB ] = BKT$M_LASTBKT;
;  386        0447  3
;  387        0448  3      ! data BDB saved for index formatting routines
;  388        0449  3      !
;  389        0450  3      IRAB [ IRB$L_CURBDB ] = .BDB;
;  390        0451  3      IRAB [ IRB$L_NXTBDB ] = 0
;  391        0452  3
;  392        0453  2      END;                          ! end of local definition of BDB + BUCKET
;  393        0454  2
;  394        0455  2      ! Now make index levels
;  395        0456  2      !
;  396        0457  2      DECR I FROM 1 TO 0 DO
;  397        0458  3          BEGIN
;  398        0459  3
;  399        0460  3          ! Choose area to use.
;  400        0461  3          !
;  401        0462  4          BEGIN
;  402        0463  4
;  403        0464  4          LOCAL
;  404        0465  4              AREA_NO;
;  405        0466  4
;  406        0467  4          IF .I EQL 0
;  407        0468  4          THEN
;  408        0469  5              BEGIN
;  409        0470  5
;  410        0471  5              IF .IDX_DFN [ IDX$B_LANUM ] EQL 0
;  411        0472  5              THEN
;  412        0473  5                  EXITLOOP;       ! There are not 2 levels of index if exitloop
;  413        0474  5
;  414        0475  5              AREA_NO = .IDX_DFN [ IDX$B_IANUM ];
;  415        0476  5
;  416        0477  5              IF .IDX_DFN [ IDX$B_LANUM ] EQL .AREA_NO<0, 8>
;  417        0478  5              THEN
;  418        0479  5                  EXITLOOP;
;  419        0480  5
;  420        0481  5              END
;  421        0482  4          ELSE
;  422        0483  5              BEGIN
;  423        0484  5              AREA_NO = .IDX_DFN [ IDX$B_LANUM ];
;  424        0485  5
;  425        0486  5              IF .AREA_NO EQL 0
;  426        0487  5              THEN
;  427        0488  5                  AREA_NO = .IDX_DFN [ IDX$B_IANUM ];
;  428        0489  5
;  429        0490  4              END;
;  430        0491  4
;  431        0492  4          ! Pickup information needed from lower level bucket before writing it
;  432        0493  4          ! out.
;  433        0494  4          !
;  434        0495  5          BEGIN
;  435        0496  5
;  436        0497  5          GLOBAL REGISTER
;  437        0498  5              R_BDB_STR;
;  438          0499  5
;  439          0500  5              BDB = .IRAB [ IRB$L_CURBDB ];
;  440          0501  5              IRAB [ IRB$L_CURBDB ] = 0;
;  441          0502  5              LOWER_VBN = .BDB [ BDB$L_VBN ];
;  442          0503  5              LEVEL = .BBLOCK [ .BDB [ BDB$L_ADDR ],BKT$B_LEVEL ] + 1;
;  443          0504  5              BDB [ BDB$V_DRT ] = 1;
;  444          0505  5              BDB [ BDB$V_VAL ] = 1;
;  445          0506  5
;  446        P 0507  5              RETURN_ON_ERROR( RM$RELEASE( RLS$M_WRT_THRU ),
;  447        P 0508  5                          BEGIN
;  448        P 0509  5                          RELEASE_KEYDESC()
;  449          0510  6                          END )
;  450          0511  6
;  451          0512  4              END;                            ! end of global register definition
;  452          0513  4
;  453          0514  4              ! Allocate and do basic formatting of one index bucket
;  454          0515  4              !
;  455        P 0516  4              RETURN_ON_ERROR( RM$AL_FRMT_BKT( .AREA_NO,
;  456        P 0517  4                                              .IDX_DFN [ IDX$B_IDXBKTSZ ] * 512 ),
;  457        P 0518  4                          BEGIN
;  458        P 0519  4                          RELEASE_KEYDESC()
;  459          0520  5                          END )
;  460          0521  5
;  461          0522  3              END;                                    ! end of local area_no
;  462          0523  3
;  463          0524  4              BEGIN
;  464          0525  4
;  465          0526  4              LOCAL
;  466          0527  4                  BUCKET      : REF BBLOCK;
;  467          0528  4
;  468          0529  4              BUCKET = .BBLOCK [ .IRAB [ IRB$L_NXTBDB ],BDB$L_ADDR ];
;  469          0530  4              BUCKET [ BKT$L_NXTBKT ] = .BBLOCK [ .IRAB [ IRB$L_NXTBDB ],BDB$L_VBN ];
;  470          0531  4              BUCKET [ BKT$B_LEVEL ] = .LEVEL;
;  471          0532  4              BUCKET [ BKT$V_LASTBKT ] = 1;
;  472          0533  4
;  473          0534  4              ! Switch IRAB BDB which describes new index bucket
;  474          0535  4              !
;  475          0536  4              IRAB [ IRB$L_CURBDB ] = .IRAB [ IRB$L_NXTBDB ];
;  476          0537  4              IRAB [ IRB$L_NXTBDB ] = 0;
;  477          0538  4
;  478          0539  4              ! Format an index entry
;  479          0540  4              !
;  480          0541  5              BEGIN
;  481          0542  5
;  482          0543  5              GLOBAL REGISTER
;  483          0544  5                  R_REC_ADDR;
;  484          0545  5
;  485          0546  5              REC_ADDR = .BUCKET + BKT$C_OVERHDSZ;
;  486          0547  5              RM$MAK_IDX_REC(.BUCKET);
;  487          0548  4              END;                                    ! of bdb_str and rec_addr
;  488          0549  3              END;                                    ! end local def of bucket
;  489          0550  2              END;                                    ! end DECR I
;  490          0551  2
;  491          0552  2          ! set root bucket indicator
;  492          0553  2          !
;  493          0554  3          BEGIN
;  494          0555  3
;   495           0556   3      LOCAL
;   496           0557   3          BUCKET  : REF BBLOCK;
;   497           0558   3
;   498           0559   3      BUCKET = .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$L_ADDR];
;   499           0560   3      BUCKET[BKT$B_BKTCB] = .BUCKET[BKT$B_BKTCB] OR BKT$M_ROOTBKT
;   500           0561   2      END;
;   501           0562   2
;   502           0563   2      ! save information about the root in the disk key descriptor and write out
;   503           0564   2      ! root
;   504           0565   2      !
;   505           0566   3      BEGIN
;   506           0567   3
;   507           0568   3      GLOBAL REGISTER
;   508           0569   3          R_BDB_STR;
;   509           0570   3
;   510           0571   3      LOCAL
;   511           0572   3          DISK_KEY_DESC   : REF BBLOCK;
;   512           0573   3
;   513           0574   3      BDB = .IRAB[IRB$L_CURBDB];
;   514           0575   3      IRAB[IRB$L_CURBDB] = 0;
;   515           0576   3      BDB[BDB$V_DRT] = 1;
;   516           0577   3      BDB[BDB$V_VAL] = 1;
;   517           0578   3      DISK_KEY_DESC = .IRAB[IRB$L_MIDX_TMP3];
;   518           0579   3      DISK_KEY_DESC[KEY$L_ROOTVBN] = .BDB[BDB$L_VBN];
;   519           0580   3      DISK_KEY_DESC[KEY$B_ROOTLEV] = .BBLOCK[.BDB[BDB$L_ADDR], BKT$B_LEVEL];
;   520           0581   3
;   521         P 0582   3      RETURN_ON_ERROR (RM$RELEASE(RLS$M_WRT_THRU),
;   522         P 0583   3          BEGIN
;   523         P 0584   3          RELEASE_KEYDESC()
;   524           0585   3          END);
;   525           0586   3
;   526           0587   3      ! Now update key descriptor and write it out
;   527           0588   3      !
;   528           0589   3      DISK_KEY_DESC[KEY$B_FLAGS] = .DISK_KEY_DESC[KEY$B_FLAGS] AND NOT KEY$M_INITIDX;
;   529           0590   3      BDB = .IRAB[IRB$L_LOCK_BDB];
;   530           0591   3      IRAB[IRB$L_LOCK_BDB] = 0;
;   531           0592   3      RM$MAKSUM(.BDB[BDB$L_ADDR]);
;   532           0593   3      BDB[BDB$V_DRT] = 1;
;   533           0594   3      BDB[BDB$V_VAL] = 1;
;   534           0595   3
;   535           0596   3      RETURN_ON_ERROR (RM$RELEASE(RLS$M_WRT_THRU));
;   536           0597   3
;   537           0598   3      ! Now call read key descriptor inorder to update the in core descriptor
;   538           0599   3      ! and verify changes got to disk.
;   539           0600   3      !
;   540           0601   3      IRAB[IRB$V_NEW_IDX] = 1;
;   541           0602   3
;   542           0603   4      RETURN_ON_ERROR (RM$KEY_DESC(.IDX_DFN[IDX$B_KEYREF]))
;   543           0604   4
;   544           0605   2      END;                                    ! end global register r_bdb_str
;   545           0606   2      RETURN 1;
;   546           0607   2
;   547           0608   1      END;
```

```
                  007C    8F  BB 00000 RM$MAKE_INDEX::
                                              PUSHR    #^M<R2,R3,R4,R5,R6>                         ; 0329
       42   A9              08  88 00004      BISB2    #8, 66(IRAB)                                ; 0383
       40   A9              01  90 00008      MOVB     #1, 64(IRAB)                                ; 0387
            7E       21     A7  9A 0000C      MOVZBL   33(IDX_DFN), -(SP)                          ; 0389
                          0000G 30 00010      BSBW     RM$KEY_DESC                                 ;
            5E              04  C0 00013      ADDL2    #4, SP                                      ;
            03              50  E8 00016      BLBS     STATUS, 1$                                  ;
                           016A 31 00019      BRW      15$                                         ;
            54     0084     C9  D0 0001C 1$:  MOVL     132(IRAB), BDB                              ; 0398
  05   1C   A7              04  E0 00021      BBS      #4, 28(IDX_DFN), 2$                         ; 0400
                            C2  10 00026      BSBB     RELEASE_KEYDESC                             ; 0403
                           0158 31 00028      BRW      14$                                         ; 0404
            50       0E     A7  3C 0002B 2$:  MOVZWL   14(IDX_DFN), R0                             ; 0410
     0090   C9       18   B440  9E 0002F      MOVAB    @24(BDB)[R0], 144(IRAB)                     ;
       0A   A4              01  8A 00036      BICB2    #1, 10(BDB)                                 ; 0414
            50       17     A7  9A 0003A      MOVZBL   23(IDX_DFN), R0                             ; 0426
  7E        50              09  78 0003E      ASHL     #9, R0, -(SP)                               ;
            7E       14     A7  9A 00042      MOVZBL   20(IDX_DFN), -(SP)                          ;
                          0000G 30 00046      BSBW     RM$AL_FRMT_BKT                              ;
            5E              08  C0 00049      ADDL2    #8, SP                                      ;
            52              50  D0 0004C      MOVL     R0, STATUS                                  ;
            71              52  E9 0004F      BLBC     STATUS, 7$                                  ;
            52       3C     A9  D0 00052      MOVL     60(IRAB), BDB                               ; 0437
            50       18     A2  D0 00056      MOVL     24(BDB), BUCKET                             ; 0438
            53       1C     A2  D0 0005A      MOVL     28(BDB), VBN                                ; 0439
       08   A0              53  D0 0005E      MOVL     VBN, 8(BUCKET)                              ; 0440
            51     0090     C9  D0 00062      MOVL     144(IRAB), R1                               ; 0444
       54   A1              53  D0 00067      MOVL     VBN, 84(R1)                                 ;
       0C   A0     0100     8F  B0 0006B      MOVW     #256, 12(BUCKET)                            ; 0445
       20   A9              52  D0 00071      MOVL     BDB, 32(IRAB)                               ; 0450
                     3C     A9  D4 00075      CLRL     60(IRAB)                                    ; 0451
            55              01  D0 00078      MOVL     #1, I                                       ; 0457
                            11  12 0007B 3$:  BNEQ     5$                                          ; 0467
                     13     A7  95 0007D      TSTB     19(IDX_DFN)                                 ; 0471
                            0A  13 00080      BEQL     4$                                          ;
            56       12     A7  9A 00082      MOVZBL   18(IDX_DFN), AREA_NO                        ; 0475
            56       13     A7  91 00086      CMPB     19(IDX_DFN), AREA_NO                        ; 0477
                            0C  12 0008A      BNEQ     6$                                          ;
                            7B  11 0008C 4$:  BRB      9$                                          ; 0479
            56       13     A7  9A 0008E 5$:  MOVZBL   19(IDX_DFN), AREA_NO                        ; 0484
                            04  12 00092      BNEQ     6$                                          ; 0486
            56       12     A7  9A 00094      MOVZBL   18(IDX_DFN), AREA_NO                        ; 0488
            54       20     A9  D0 00098 6$:  MOVL     32(IRAB), BDB                               ; 0500
                     20     A9  D4 0009C      CLRL     32(IRAB)                                    ; 0501
     0088   C9       1C     A4  D0 0009F      MOVL     28(BDB), 136(IRAB)                          ; 0502
            50       18     A4  D0 000A5      MOVL     24(BDB), R0                                 ; 0503
     008C   C9       0C     A0  9A 000A9      MOVZBL   12(R0), 140(IRAB)                           ;
                   008C     C9  D6 000AF      INCL     140(IRAB)                                   ;
       0A   A4              03  88 000B3      BISB2    #3, 10(BDB)                                 ; 0505
            53              02  D0 000B7      MOVL     #2, R3                                      ; 0510
               00000000G    EF  16 000BA      JSB      RM$RELEASE                                  ;
            52              50  D0 000C0      MOVL     R0, STATUS                                  ;
            13              52  E9 000C3 7$:  BLBC     STATUS, 8$                                  ;
            50       16     A7  9A 000C6      MOVZBL   22(IDX_DFN), R0                             ; 0520
  7E        50              09  78 000CA      ASHL     #9, R0, -(SP)                               ;
```

```
                     56  DD 000CE          PUSHL   AREA_NO                                    ;
                  0000G  30 000D0          BSBW    RM$AL_FRMT_BKT                             ;
     5E              08  C0 000D3          ADDL2   #8, SP                                     ;
     52              50  D0 000D6          MOVL    R0, STATUS                                 ;
     6A              52  E9 000D9 8$:      BLBC    STATUS, 12$                                ;
     51        3C    A9  D0 000DC          MOVL    60(IRAB), R1                               ; 0529
     50        18    A1  D0 000E0          MOVL    24(R1), BUCKET                             ;
08   A0        1C    A1  D0 000E4          MOVL    28(R1), 8(BUCKET)                          ; 0530
0C   A0      008C    C9  90 000E9          MOVB    140(IRAB), 12(BUCKET)                      ; 0531
0D   A0              01  88 000EF          BISB2   #1, 13(BUCKET)                             ; 0532
20   A9              51  D0 000F3          MOVL    R1, 32(IRAB)                               ; 0536
               3C    A9  D4 000F7          CLRL    60(IRAB)                                   ; 0537
     56        0E    A0  9E 000FA          MOVAB   14(R0), REC_ADDR                           ; 0546
                     50  DD 000FE          PUSHL   BUCKET                                     ; 0547
                  0000V  30 00100          BSBW    RM$MAK_IDX_REC                             ;
     5E              04  C0 00103          ADDL2   #4, SP                                     ;
     02              55  F4 00106          SOBGEQ  I, 10$                                     ; 0457
                     03  11 00109 9$:      BRB     11$                                        ;
                   FF6D  31 0010B 10$:     BRW     3$                                         ;
     51        20    A9  D0 0010E 11$:     MOVL    32(IRAB), R1                               ; 0559
     50        18    A1  D0 00112          MOVL    24(R1), BUCKET                             ;
0D   A0              02  88 00116          BISB2   #2, 13(BUCKET)                             ; 0560
     54              51  D0 0011A          MOVL    R1, BDB                                    ; 0574
               20    A9  D4 0011D          CLRL    32(IRAB)                                   ; 0575
0A   A4              03  88 00120          BISB2   #3, 10(BDB)                                ; 0577
     55      0090    C9  D0 00124          MOVL    144(IRAB), DISK_KEY_DESC                   ; 0578
0C   A5        1C    A4  D0 00129          MOVL    28(BDB), 12(DISK_KEY_DESC)                 ; 0579
     50        18    A4  D0 0012E          MOVL    24(BDB), R0                                ; 0580
09   A5        0C    A0  90 00132          MOVB    12(R0), 9(DISK_KEY_DESC)                   ;
     53              02  D0 00137          MOVL    #2, R3                                     ; 0585
          00000000G  EF  16 0013A          JSB     RM$RELEASE                                 ;
     52              50  D0 00140          MOVL    R0, STATUS                                 ;
     08              52  E8 00143          BLBS    STATUS, 13$                                ;
                   FEA1  30 00146 12$:     BSBW    RELEASE_KEYDESC                            ;
     50              52  D0 00149          MOVL    STATUS, R0                                 ;
                     38  11 0014C          BRB     15$                                        ;
10   A5              10  8A 0014E 13$:     BICB2   #16, 16(DISK_KEY_DESC)                     ; 0589
     54      0084    C9  D0 00152          MOVL    132(IRAB), BDB                             ; 0590
             0084    C9  D4 00157          CLRL    132(IRAB)                                  ; 0591
     55        18    A4  D0 0015B          MOVL    24(BDB), R5                                ; 0592
                  0000G  30 0015F          BSBW    RM$MAKSUM                                  ;
0A   A4              03  88 00162          BISB2   #3, 10(BDB)                                ; 0594
     53              02  D0 00166          MOVL    #2, R3                                     ; 0596
          00000000G  EF  16 00169          JSB     RM$RELEASE                                 ;
     14              50  E9 0016F          BLBC    STATUS, 15$                                ;
42   A9              08  88 00172          BISB2   #8, 66(IRAB)                               ; 0601
     7E        21    A7  9A 00176          MOVZBL  33(IDX_DFN), -(SP)                         ; 0603
                  0000G  30 0017A          BSBW    RM$KEY_DESC                                ;
     5E              04  C0 0017D          ADDL2   #4, SP                                     ;
     03              50  E9 00180          BLBC    STATUS, 15$                                ;
     50              01  D0 00183 14$:     MOVL    #1, R0                                     ; 0606
             007C    8F  BA 00186 15$:     POPR    #^M<R2,R3,R4,R5,R6>                        ; 0608
                         05 0018A          RSB                                                ;

; Routine Size:  395 bytes,    Routine Base:  RM$RMS3 + 005C
;  548         0609  1
```

; R

```
:  550         0610  1 %SBTTL 'RM$MAK_IDX_REC'
:  551         0611  1 GLOBAL ROUTINE RM$MAK_IDX_REC(BUCKET): RL$RABREG_67 NOVALUE =
:  552         0612  1 !+++
:  553         0613  1 !
:  554         0614  1 !    RM$MAK_IDX_REC
:  555         0615  1 !
:  556         0616  1 !         This routine builds an index record for the high key value of
:  557         0617  1 !         any flavor index bucket.
:  558         0618  1 !
:  559         0619  1 ! CALLING SEQUENCE:
:  560         0620  1 !         RM$MAK_IDX_REC (BUCKET)
:  561         0621  1 !
:  562         0622  1 ! INPUT PARAMETERS:
:  563         0623  1 !         BUCKET - address of bucket, points to where first record goes
:  564         0624  1 !
:  565         0625  1 ! IMPLICIT INPUTS:
:  566         0626  1 !
:  567         0627  1 !         IDX_DFN          - address of in core key descriptor which needs an index
:  568         0628  1 !         IFAB             - address of internal FAB
:  569         0629  1 !         IRAB             - address of internal RAB
:  570         0630  1 !         REC_ADDR         - record address for high key
:  571         0631  1 !
:  572         0632  1 ! OUTPUT PARAMETERS:
:  573         0633  1 !         NONE
:  574         0634  1 !
:  575         0635  1 ! IMPLICIT OUTPUTS:
:  576         0636  1 !         REC_ADDR updated to point past high key
:  577         0637  1 !
:  578         0638  1 ! ROUTINE VALUE:
:  579         0639  1 !         NONE
:  580         0640  1 !
:  581         0641  1 ! SIDE EFFECTS:
:  582         0642  1 !
:  583         0643  1 !
:  584         0644  1 !--
:  585         0645  1
:  586         0646  2 BEGIN
:  587         0647  2
:  588         0648  2 MAP
:  589         0649  2     BUCKET : REF BBLOCK;
:  590         0650  2
:  591         0651  2 MACRO
:  592         0652  2     KEY_LEN     = 0,0,8,0 %,
:  593         0653  2     FRNT_CMPR   = 1,0,8,0 %;
:  594         0654  2
:  595         0655  2 GLOBAL REGISTER
:  596         0656  2     COMMON_RAB_STR,
:  597         0657  2     R_BDB,
:  598         0658  2     R_IDX_DFN_STR;
:  599         0659  2
:  600         0660  2 EXTERNAL REGISTER
:  601         0661  2     R_REC_ADDR_STR;
:  602         0662  2
:  603         0663  2 LOCAL
:  604         0664  2     SIZE;
:  605         0665  2
:  606         0666  2 ! First get the size for the VBN
```

```
!

SIZE = RM$VBN_SIZE (.IRAB [IRB$L_MIDX_TMP1]);

! Now set up the record depending on bucket flavor
!

CASE .IDX_DFN [IDX$B_IDXBKTYP] FROM IDX$C_V2_BKT TO IDX$C_NCMPIDX OF

    SET

    [IDX$C_V2_BKT]:

        ! Prologue one or two index bucket
        !    --------------------
        !    ! cntrl ! VBN ! key !
        !    --------------------
        !
        BEGIN
        (.REC_ADDR)<0,8> = .SIZE - 2;
        (.REC_ADDR)<8,.SIZE*8> = .IRAB [IRB$L_MIDX_TMP1];
        REC_ADDR = .REC_ADDR + .SIZE + 1;
        MAKE_HIGH_KY();
        END;

    [IDX$C_CMPIDX]:

        ! Prologue three compressed index bucket
        !    -----------------------------
        !    ! len ! frnt compr cnt ! key !
        !    -----------------------------
        !
        BEGIN

        LOCAL
            FIRST_VBN;

        ! First build key portion
        !
        REC_ADDR [KEY_LEN] = 1;
        REC_ADDR [FRNT_CMPR] = 0;
        REC_ADDR = CH$FILL (%X'FF', 1, .REC_ADDR + 2);

        ! Now build VBN portion
        !
        BUCKET [BKT$V_PTR_SZ] = .SIZE - 2;
        FIRST_VBN = .BUCKET + (.IDX_DFN [IDX$B_IDXBKTSZ] * 512) - 4;
        (.FIRST_VBN - .SIZE) <0,.SIZE^3> = .IRAB [IRB$L_MIDX_TMP1];

        ! Insert the 'end of freespace' pointer
        !
        (.FIRST_VBN)<0,16> = .FIRST_VBN - .SIZE - .BUCKET - 1;
        END;

    [IDX$C_NCMPIDX]:

        ! Prologue three non-compressed index record
```

```
;  664          0724  2             !       -------
;  665          0725  2             !       ! key !
;  666          0726  2             !       -------
;  667          0727  2             !
;  668          0728  3             BEGIN
;  669          0729  3             LOCAL
;  670          0730  3                 FIRST_VBN;
;  671          0731  3
;  672          0732  3             MAKE_HIGH_KY ();
;  673          0733  3
;  674          0734  3             ! Now build VBN portion
;  675          0735  3             !
;  676          0736  3             BUCKET [BKT$V_PTR_SZ] = .SIZE - 2;
;  677          0737  3             FIRST_VBN = .BUCKET + (.IDX_DFN [IDX$B_IDXBKTSZ] * 512) - 4;
;  678          0738  3             (.FIRST_VBN - .SIZE) <0,.SIZE^3> = .IRAB [IRB$L_MIDX_TMP1];
;  679          0739  3
;  680          0740  3             ! Fill in the 'end of freespace' pointer
;  681          0741  3             !
;  682          0742  3             (.FIRST_VBN)<0,16> = .FIRST_VBN - .SIZE - .BUCKET - 1;
;  683          0743  2             END;
;  684          0744  2
;  685          0745  2         TES;
;  686          0746  2
;  687          0747  2 ! Now fill in the free space pointer
;  688          0748  2 !
;  689          0749  2 BUCKET [BKT$W_FREESPACE] = (.REC_ADDR - .BUCKET)<0,16>;
;  690          0750  1 END;
; INFO#250              L1:0669
; Referenced REGISTER symbol IRAB is probably not initialized



                                         0F94    8F  BB 00000 RM$MAK_IDX_REC::
                                                                   PUSHR   #^M<R2,R4,R7,R8,R9,R10,R11>              ; 0611
                                         0088    C9  DD 00004      PUSHL   136(IRAB)                                ; 0669
                                            0000G 30 00008         BSBW    RM$VBN_SIZE                              ;
                                 5E          04  C0 0000B          ADDL2   #4, SP                                   ;
                                 54          50  D0 0000E          MOVL    R0, SIZE                                 ;
                                 52      FE  A4  9E 00011          MOVAB   -2(R4), R2                               ; 0686
                      02         00      28  A7  8F 00015          CASEB   40(IDX_DFN), #0, #2                      ; 0674
                    0026       001E        0006     0001A 1$:      .WORD   2$-1$,-                                  ;
                                                                           3$-1$,-                                  ;
                                                                           4$-1$                                    ;
                                 66          52  90 00020 2$:      MOVB    R2, (REC_ADDR)                           ; 0686
                      50         54          03  78 00023          ASHL    #3, SIZE, R0                             ; 0687
           66         50         08      0088 C9 F0 00027          INSV    136(IRAB), #8, R0, (REC_ADDR)            ;
                                 56      01 A446 9E 0002E          MOVAB   1(SIZE)[REC_ADDR], REC_ADDR              ; 0688
                                           FDE3  30 00033          BSBW    MAKE_HIGH_KY                             ; 0689
                                             39  11 00036          BRB     6$                                       ; 0674
                                 86          01  B0 00038 3$:      MOVW    #1, (REC_ADDR)+                          ; 07C5
                                 86          01  8E 0003B          MNEGB   #1, (REC_ADDR)+                          ; 0708
                                             03  11 0003E          BRB     5$                                       ; 0712
                                           FDD6  30 00040 4$:      BSBW    MAKE_HIGH_KY                             ; 0732
                                 51      20  AE  D0 00043 5$:      MOVL    BUCKET, R1                               ; 0736
     0D    A1         02         03          52  F0 00047          INSV    R2, #3, #2, 13(R1)                       ;
```

```
                                    57      16   A7  9A 0004D        MOVZBL  22(IDX_DFN), R7                                     ; 0737
                       57           57           09  78 00051        ASHL    #9, R7, R7                                          :
                                    50    FC A741    9E 00055        MOVAB   -4(R7)[R1], FIRST_VBN                               :
                       58           50           54  C3 0005A        SUBL3   SIZE, FIRST_VBN, R8                                 ; 0738
                       57           54           03  78 0005E        ASHL    #3, SIZE, R7                                        :          ; R
            68         57           00    0088   C9  F0 00062        INSV    136(IRAB), #0, R7, (R8)                             :
                       51           58           51  C3 00069        SUBL3   R1, R8, R1                                          ; 0742
                       60           51           01  A3 0006D        SUBW3   #1, R1, (FIRST_VBN)                                 :
                                    50      20   AE  D0 00071 6$:    MOVL    BUCKET, R0                                          ; 0749
                  04   A0           56           50  A3 00075        SUBW3   R0, REC_ADDR, 4(R0)                                 :
                                          0F94   8F  BA 0007A        POPR    #^M<R2,R4,R7,R8,R9,R10,R11>                         ; 0750
                                                     05 0007E        RSB                                                         :

; Routine Size:  127 bytes,    Routine Base:  RM$RMS3 + 01E7


;  691          0751  1
;  692          0752  1 END
;  693          0753  1
;  694          0754  0 ELUDOM





;                                   PSECT SUMMARY
;
;       Name                            Bytes                           Attributes
;
;  RM$RMS3                                    614  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        70         2        154          00:00.4


; Information:   1
; Warnings:      0
; Errors:        0


;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3MAKIDX/OBJ=OBJ$:RM3MAKIDX MSRC$:RM3MAKIDX/UPDATE=(ENH$:RM3MAKIDX)

; Size:          614 code + 0 data bytes
; Run Time:         00:16.5
; Elapsed Time:    00:43.0
; Lines/CPU Min:    2745
; Lexemes/CPU-Min: 18826
; Memory Used:  170 pages
; Compilation Complete
```

0325 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY